APLICACIÓN DE TÉCNICAS MATHEURISTICAS AL PROBLEMA DE RUTEO ÓPTIMO DE VEHÍCULOS CONSIDERANDO UNA METODOLOGÍA DE REDUCCIÓN DEL ESPACIO DE BÚSQUEDA.

Luis Hernando Martínez Rubio
Cod: 1087994883

Director
PhD. Mauricio Granada E.

Universidad Tecnológica De Pereira
Facultad De Ingenierías: Eléctrica, Electrónica, Física Y Ciencias De La Computación
Programa De Maestría En Ingeniería Eléctrica
Pereira
2015

Dedicatoria:

A Dios, a mis padres, mi novia, mis hermanas y al resto de mi familia.

Agradecimientos:

A los profesores del programa de Maestría en Ingeniería eléctrica de la Universidad Tecnológica de Pereira, quienes aportaron en mi desarrollo profesional y de este trabajo. A mis amigos, compañeros y colegas por su colaboración, por último al director y asesor de este trabajo, Mauricio Granada E. por su completa colaboración y asesoría en el desarrollo de este proyecto.

Contenido

Tabla de Figuras

1 Problema de Ruteo de Vehículos

1.1 Introducción

Las empresas de provisión y distribución de bienes buscan ofrecer un servicio óptimo, este consiste en ofrecer el mejor servicio con los costos de operación más bajos posibles. Los costos de operación están asociados a la mano de obra, costos de utilización de vehículos, entre otros. Algunas empresas ven el tiempo como un factor monetario, cuando se debe cumplir con la entrega o colecta de un bien en una franja de tiempo determinado. Otros ven la necesidad de que la programación de las rutas de cada vehículo sea lo más equitativa posible en cuanto a la capacidad de carga del vehículo. Por tal motivo algunas empresas invierten en softwares o investigaciones que conlleven a planificar las mejores estrategias de operación de su sistema.

Un gran número de aplicaciones del mundo real, tanto en Norte América como en Europa, han demostrado ampliamente que el uso de procedimientos informáticos para la planificación del proceso de distribución produce un ahorro sustancial (estimado entre el 5% y el 20%) en los costos globales de transporte [1]. Una disminución de los costos de operación en los problemas reales, se traduce en un aumento del beneficio de la empresa o en una disminución de los costos finales del producto. La investigación de operaciones busca optimizar el beneficio de las empresas distribuidoras, valiéndose de modelos matemáticos que describen el proceso que se desea mejorar, basados en un objetivo previamente definido. Estos modelos posteriormente son resueltos mediante técnicas exactas, metaheurísticas y técnicas híbridas conocidas como matheurísticas.

El problema de ruteo de vehículos VRP, es una formulación del tipo lineal entero mixto. El uso de técnicas exactas en este tipo de problemas está restricto a la complejidad del modelo matemático que lo describe, como por ejemplo, el número de variables discretas o el número de restricciones. Estos problemas generan una explosión combinatorial que se refleja en los altos tiempos de cómputo para obtener la solución óptima global, adicionalmente dichos tiempos crecen exponencialmente con el tamaño del problema, razón por la cual, en términos de la complejidad computacional se clasifica dentro de la categoría NP-Hard. Lo anterior, explica el interés creciente en la utilización de las técnicas heurísticas y metaheurísticas, que si bien no tienen un sustento matemático como las técnicas exactas, su uso se sustenta o justifica por la calidad de los resultados obtenidos, beneficiosos para algunos sectores de producción e investigación. La literatura especializada en dichos problemas, emplean técnicas como algoritmos evolutivos o genéticos, colonia de hormigas, algoritmo de cumulo

de partículas, etc. Muchos de los trabajos emplean las mismas técnicas, la diferencia radica en la forma como se trata el problema, como por ejemplo la estrategia o heurística empleada para construir una solución del problema. Algunos de estos algoritmos son potenciados con algoritmos de búsqueda local, por lo que las anteriores técnicas son empleadas como mecanismo de exploración en el espacio de soluciones y la búsqueda local actúa como un mecanismo de explotación.

La mayor desventaja de estos algoritmos es el hecho de que no se garantiza que la solución obtenida sea el óptimo global del problema, ya que exploran solo una parte del espacio de soluciones. Adicionalmente si no hay una buena parametrización de los factores que controlan el algoritmo, se puede caer fácilmente en soluciones de baja calidad u óptimos locales. Algoritmos como búsqueda tabú granular, el cual es una técnica de búsqueda local que restringe ciertas soluciones para determinadas variables, tiene como finalidad evitar caer fácilmente en óptimos locales. Restringir el espacio de soluciones disminuye el espacio de búsqueda y a su vez el tiempo de computo, sin embargo se debe ser cautelosos al restringir dicho espacio, ya que es posible que el óptimo global quede por fuera de éste.

Este trabajo propone el uso de técnicas metaheurísticas, el cual se mueve por un espacio de soluciones restringido. En una primera etapa, las soluciones construidas contemplan el uso de ciertos arcos, seleccionados de acuerdo a una estrategia basada en un criterio de dominancia utilizado en las técnicas de optimización multiobjetivo. En esta etapa se construye un conjunto de rutas factibles que servirán como información de entrada a una segunda etapa. La metodología culmina con la implementación de un modelo de programación lineal entero que utiliza el conjunto de rutas para encontrar una solución de mejor calidad. Este modelo corresponde a uno de partición de conjuntos que puede ser resuelto de forma exacta a través de una metodología de ramificación y poda (B&B).

1.2 Definición del problema (CVRP)

El problema de ruteo óptimo de vehículos (VRP, por su acrónimo en inglés), es un problema referente en la optimización matemática. Generalmente, su modelo es formulado como uno de programación lineal entera mixta, el cual resulta ser intuitivo y puede ser generalizado para aplicarse a otros sistemas, como por ejemplo el problema de planeamiento de sistemas eléctricos. Sin embargo, la complejidad computacional involucrada en la solución del modelo matemático del VRP, hace que el uso de técnicas exactas este definido por el del problema.

El VRP consiste en la determinación óptima de un conjunto de rutas programadas para una flota de vehículos que sirven un conjunto dado de clientes, y es uno de los más estudiados e importantes problemas de optimización combinatoria [1].

El servicio de distribución de bienes consiste en programar el recorrido realizado por los choferes de una flota de vehículos que parten de un depósito y se desplazan a través de las redes viales, con el fin de entregar en cada uno de los puntos de servicio la cantidad del bien demandado. En el caso en que se asume que la capacidad de carga de los vehículos es homogénea el problema se define como CVRP. En este caso, los vehículos no deben exceder la capacidad de carga prestablecida. Lo que se desea, es disminuir los costos de la operación que implica la distribución de los bienes. Estos costos están asociados con la mano de obra, los costos de utilización de los vehículos, penalidades o infracciones en los tiempos de entrega a los clientes entre otros. Estos son denominados objetivos del problema y el modelo matemático puede involucrar uno o varios de estos objetivos, dentro de una misma función objetivo (mono-objetivo) o en múltiples objetivos (multi-objetivo). Para el caso básico del CVRP se desea minimizar la distancia total recorrida por los vehículos operados, el cual es directamente proporcional al costo de la operación.

En problemas reales determinar la distancia que hay de un punto a otro con base a las redes viales es algo poco práctico, debido a la cantidad de opciones que dicha red puede ofrecer. Razón por la cual se suele representar con las distancias euclidianas, aunque esto puede traer algunas consecuencias en términos de las distancias realmente recorridas por las flotas.

En este proyecto se propone resolver eficientemente el CVRP, presentando una novedosa técnica de reducción del espacio de búsqueda y una metodología de solución basada en optimización por colonia de hormigas (ACO).

1.3 Antecedentes del problema

El problema de ruteo de vehículos fue propuesto en Dantzig y Ramser, en 1959 [2], con el propósito de ruteo de tanques distribuidores de gasolina a sus estaciones de servicio.

A partir de dicha formulación, a lo largo de la historia son diversos los trabajos de investigación referentes al VRP, el crecimiento de dichos trabajos es de orden exponencial, tal como se ilustra en

la Figura 1.1 y Figura 1.2, proporcionadas por la referencia [3], el cual muestra el histórico desde 1954, hasta el 2006.

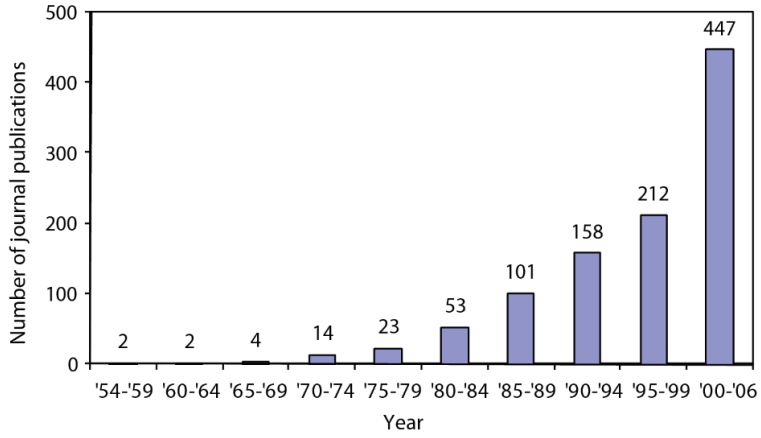


Figura 1.1. Número de Artículos VRP publicados en revistas indexadas desde 1954 hasta 2006. Tomado de [3].

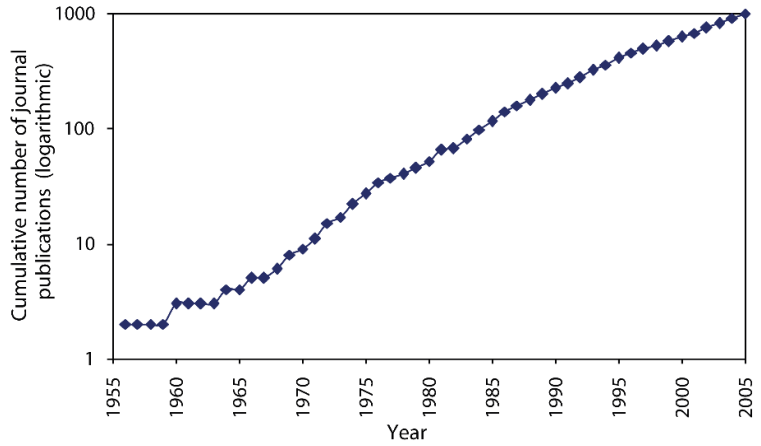


Figura 1.2. Número acumulativo de artículos de VRP para el periodo 1956-2005. Tomado de [3].

Dicha referencia proporciona una taxonomía, el cual muestra de manera minuciosa una clasificación de los trabajos referentes al VRP. En este proyecto se mencionan algunas cuantas referencias, las cuales se consideran fundamentales para los propósitos de este trabajo.

Los modelos del VRP son del tipo lineal entero mixto, en donde la complejidad del problema surge tanto en la utilización de variables enteras (variables complicantes) como en la incorporación de restricciones asociadas a la eliminación de subtours (restricciones complicantes). Este tipo de

problemas es considerado NP-HARD y ha recurrido a el uso de técnicas exactas, heurísticas, metaheuristicas o metodologías hibridas.

Clarke y Wright [4], exponen un algoritmo para la solución del problema del ruteo de vehículos con carga, que a menudo se llama el problema de ruteo de vehículos clásicos. Este algoritmo es denominado algoritmo de ahorros y es una heurística muy popular, ya que proporciona soluciones de alta calidad en bajo tiempo de cómputo. Este algoritmo es muy empleado para obtener soluciones iniciales en métodos más complejos.

En [5] se presenta un método que emplea dos metodologías para obtener soluciones al problema del VRP, mediante el árbol de mínima expansión central de grado k (K-CDT) y el método de las q-rutas. La metodología culmina con un algoritmo branch and bound que emplea las metodologías anteriores como solución de cada nodo o sub-problema generado. Otros algoritmos de ramificación y poda son propuestos en [6], [7], para este último se emplea el Branch and Cut and Price, adicionalmente emplea la metodología de q-rutas. 2 formulaciones son descritas, una basada en Set Partitioning y otra basada en arco-demanda.

El problema de los métodos de arborecencia es la cantidad de sub-problemas que se puedan generar, el cual se ve reflejado en el tiempo de cómputo, adicionalmente la deficiencia crece con el tamaño del problema, para ciertos problemas no se llega a la solución óptima por límites de tiempo o de memoria. Las heurísticas y metaheurísticas ofrecen soluciones rápidas, aunque no son el óptimo global, sin embargo dichas soluciones son de buena calidad, mediante, algunos métodos han logrado llegar al óptimo global del problema.

En [8] se presenta un algoritmo genético para resolver el problema del CVRP, esta metodología es comparada con la técnica golosa de búsqueda aleatoria adaptativa (GRASP). El algoritmo genético emplea una población factible, mediante la permutación del vector de nodos o clientes, como mecanismo de cruzamiento se usa “Partially Mapping Crossover” (PMX), el cual evita que el hijo generado sea infactible y conserva la información genética de los padres (orden de cada nodo en los vectores permutados), como instancias emplean las categorías A y B propuestas en [9]. En sus resultados se observa que GRASP supera al AG en el promedio de los costos obtenidos, sin embargo el algoritmo genético es superior en la categoría B.

GRASP es una técnica que emplea 2 fases, la primera consiste en construir soluciones de forma aleatoria, la segunda consiste en la búsqueda local. En [10] se aborda el problema del VRP mediante

una modificación de esta técnica, denominada Multiple Phase NeighborhoodSearch-GRASP (MPNS-GRASP), esta implementa un criterio de parada basado en relajación lagrangeana, también emplea optimización del sub-gradiente. En la fase de búsqueda local implementa una metodología de círculos que restringe los movimientos de la búsqueda.

En [11] se propone una metodología híbrida para resolver el problema del CVRP, dicha metodología involucra cúmulo de partículas para resolver problemas de variables discretas (CPSO) y el algoritmo de recocido simulado (SA). Adicionalmente se adopta la metodología de agrupar primero y luego generar la ruta (cluster first and route second), de tal forma que se emplea el algoritmo CPSO para asignar a cada cliente un vehículo, teniendo en cuenta la capacidad de este, posteriormente SA se encarga de determinar el orden del recorrido de los vértices (ruta) de cada vehículo. Durante una iteración se toman las 3 mejores partículas (soluciones) y se realiza una búsqueda local. Esta consiste en generar de manera aleatoria dos rutas, para así seleccionar un cliente de cada una de estas e intercambiarlas, sin que se reboce la capacidad del vehículo. Los autores afirman que se obtienen soluciones a dicho problema en tiempos de cómputo razonables. De manera similar [12] propone una técnica compuesta por la técnica de agrupamiento K-Means y Artificial Fish-Swarm Algorithm (AFSA), dicha combinación es denominada KMAFA. Para este caso se tiene que el número de agrupamientos K, es igual al número de vehículos, los resultados obtenidos son comparados con técnicas como colonia de hormigas y algoritmo genético hibrido, de tal manera que los autores concluyen que dichos resultados son mejores a las 2 últimas mencionadas.

Una propuesta de reducción en el espacio de búsqueda es presentada en [13], donde se emplea una metodología la cual genera un anillo que recorre los vértices más alejados al depósito, dicho anillo contiene en su interior todos los vértices del problema, a partir de ahí se genera una estrategia que parte del depósito y conecta a los nodos del interior con los nodos que conforman el anillo, posteriormente se deben podar ciertos arcos de tal manera que queden 2k arcos.

Los métodos de agrupamiento, pueden considerarse como métodos que reducen el espacio de búsqueda, ya que las soluciones obtenidas se restringen a los conjuntos de vértices agrupados. Los métodos más comunes consisten en agrupar los vértices en sub-conjuntos y posteriormente generar la ruta de cada uno de estos. La propuesta de la referencia [14] consiste en un algoritmo denominado “route-nearest neighbor” (RNN), el cual es contrastado con el algoritmo de vecino más cercano “Nearest Neighbor” (NN), la diferencia radica en que la primera metodología busca el próximo nodo más cercano a la ruta, mientras que NN busca el nodo más cercano al nodo actual. La metodología es

probada en un sistema real de gran escala, correspondiente a una distribuidora de cigarrillos, con 6772 clientes, los cuales son divididos en 5 regiones o sub-problemas, luego, cada sub-problema es resuelto mediante la metodología propuesta por los autores. De manera similar, [15] trabaja el problema a gran escala e implementan una técnica de 2 pasos, en el primer paso se emplea la técnica k-means para una clasificación geográfica de los clientes, en el segundo paso se balancea los grupos obtenidos, de tal manera que el número de usuarios por grupo sea similar, en este caso se emplea una instancia de 1882 clientes.

El algoritmo de colonia de hormigas propuesto por Dorigo y Stützle en [16] ha tenido gran uso en problemas de optimización que involucran grafos, este algoritmo es una técnica inspirada en el comportamiento de la colonia frente a la búsqueda de alimento, las hormigas se orientan en su espacio de búsqueda mediante la feromona depositada por las hormigas que han encontrado el alimento. En cuanto al problema del VRP la cantidad de feromona depositada es proporcional a la calidad de la solución. Una de las mayores dificultades de este algoritmo, es que puede caer rápidamente en soluciones de baja calidad, inclusive el uso de fenómeno de la evaporación en la matriz de feromonas no es suficiente para evitar este hecho. Algunos autores potencian el algoritmo de colonia de hormigas proponiendo una serie de estrategias.

En el caso de [17], se presenta un algoritmo de colonia de hormigas híbrido donde, se implementa un algoritmo estocástico de búsqueda local (SLSACS), en esta etapa solo algunos nodos son seleccionados de manera aleatoria para obtener su probabilidad de transición. El algoritmo es probado en 14 instancias, correspondientes a las instancias de Solomon. Dichos resultados muestran como la solución obtenida con la metodología propuesta por el autor supera las mejores soluciones reportadas previamente para las 12 instancias.

En [18] se implementa un algoritmo de colonia de hormigas que implementan 2 mecanismos que evitan caer fácilmente a óptimos locales, dichos mecanismos permiten explorar soluciones nuevas, el primer mecanismo afecta la feromona, mientras que el segundo mecanismo afecta las probabilidades de transición.

Otros proponen emplear múltiples colonias de hormigas o crear sub-grupos dentro de una colonia, cada sub-grupo tiene criterios o hábitos de selección diferente, esto garantiza cierta diversidad en las soluciones obtenidas por cada sub-grupo o colonia. En [19] se plantea un algoritmo de colonia de hormigas de comportamiento hibrido, el algoritmo utiliza 3 tipos de grupos de hormigas cuyas probabilidades de transición se calculan de manera diferente, la primera consiste en la manera

tradicional, en el que la probabilidad de la transición de *i* a *j* depende de la feromona depositada en *ij* y en la distancia *ij*. El segundo grupo de hormigas la probabilidad de transición depende de los 2 factores anteriores, adicionalmente, se introduce también el ángulo entre *0i* y *0j* (dónde *0* corresponde al depósito). Por último, en el tercer grupo de hormigas, la probabilidad de transición es un número aleatorio entre 0 y 1 siempre y cuando el nodo seleccionado no haya sido visitado, de lo contrario la probabilidad de transición es 0. El algoritmo es probado con los siguientes parámetros. Número de hormigas por grupo: *m1= 60, m2=30, m3=10,* $\alpha = 1, \beta = 5, \rho = 0.05, \tau_0 = 0.1$, número máximo de iteraciones *Imax =30*. El algoritmo es probado en 4 instancias de Christofides y Eilon. Los autores muestran que para las instancias probadas, las soluciones obtenidas son mejores en comparación con el método de colonia de hormigas tradicional.

Los algoritmos heurísticos o meta-heurísticos se caracterizan por generar soluciones a lo largo de su ejecución. Algunos métodos aprovechan algunas de estas soluciones para obtener nuevas soluciones, esto puede verse como una combinación lineal de las soluciones empleadas, como es el caso de "Path relinking" (re-encanedamiendo de trayectorias), el cual es un mecanismo muy empleado en el algoritmo de búsqueda tabú y en al algoritmo GRASP. Otros autores recurren al uso de técnicas exactas, en este caso el conjunto de soluciones puede verse como una combinación lineal convexa que genera un sub-conjunto del espacio de soluciones donde posiblemente se encuentre el óptimo global del problema. [20] aplica el modelo de "set partitioning" (SP) con generación de columnas. Este método emplea un paquete de rutas que serán ensambladas de tal manera que el costo sea mínimo y se respete las restricciones impuestas por el problema del VRP. El conjunto de soluciones son obtenidas mediante un algoritmo de búsqueda local denominado "ILS-RVND heuristic" descrito en las referencias [21], [22] dicho trabajo es aplicado a diversas extensiones del VRP, como, instancias del VRP capacitado, con colecta y entrega simultánea, mixto con colecta y entrega, multi-depósito mixto con colecta y entrega.

Finalmente, para una mayor información del lector, una recopilación destacable de las metodologías empleadas para el problema de ruteo de vehículos es desarrollada por los autores Toht y Vigo, en su libro titulado "The vehicle routing problem" [1]. Diversos modelos y metodologías planteados por diversos autores son recopiladas en dicho libro, el cual es considerado en este trabajo como un BUEN referente al VRP.

1.4 Modelos matemáticos del problema

El problema del VRP, consiste en establecer la ruta de *m* vehículos que parten de un depósito, estos deben entregar unidades de mercancía a los *n* clientes asociados de dicho depósito, el objetivo es minimizar los costos de operación que por lo general están representados por la distancia recorrida de los vehículos.

Cada cliente es visitado una sola vez (grado del vértice igual a 2):

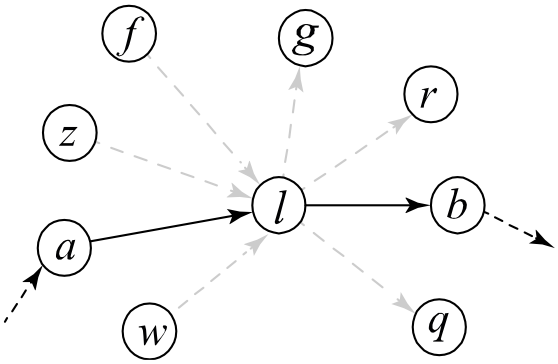


Figura 1.3. Grado del vértice o de los clientes.

Del depósito salen y llegan m rutas (grado del depósito igual a *2m*):

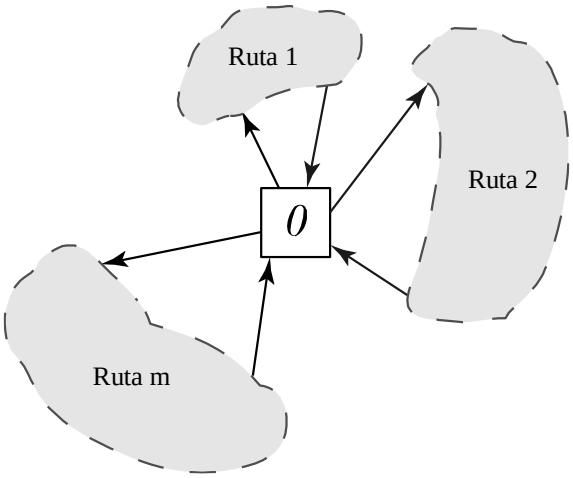


Figura 1.4. Grado del depósito.

1.4.1 Modelo Tradicional

En la referencia [1], se muestra el siguiente modelo matemático que describe el problema de ruteo de vehículos:

$$\min_{x}\left\{\sum_{i\in V}\sum_{j\in V}c_{ij}x_{ij}\right\} \tag{1.1}$$

s.a:

$$\sum_{i\in V}x_{ij}=1 \quad \forall i\in V\setminus\{0\} \tag{1.2}$$

$$\sum_{j\in V}x_{ij}=1 \quad \forall i\in V\setminus\{0\} \tag{1.3}$$

$$\sum_{i\in V}x_{i0}=K \tag{1.4}$$

$$\sum_{j\in V}x_{0j}=K \tag{1.5}$$

$$\sum_{i\notin S}\sum_{j\in S}x_{ij}>r(S) \quad \forall S\subseteq V\setminus\{0\}\vee S\neq\varnothing \tag{1.6}$$

$$x_{ij}\in\{0,1\} \quad \forall i,j\in A \tag{1.7}$$

La complejidad de este modelo radica en la cantidad de cortes que se deben generar para obtener una solución factible al problema (eliminación de sub-tours).

1.4.2 Modelo del CVRP como un problema de flujo en red

El problema de flujo en red consiste en distribuir el flujo que parte de un nodo fuente a través de un conjunto de redes hasta el nodo sumidero. El flujo por cada red no debe exceder la capacidad de estos.

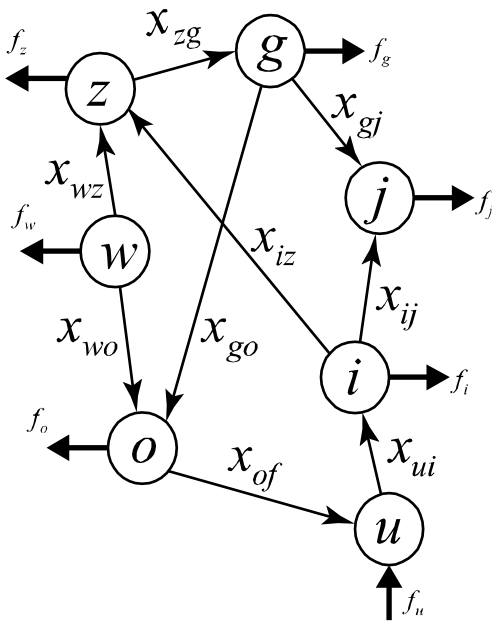


Figura 1.5. Flujo en red.

Para el problema de flujo en red se desea minimizar el costo del flujo por la red, el modelo se muestra en (1.8)-(1.11).

$$z = \min\left\{\sum_{i}\sum_{\substack{j \\ i\neq j}} r_{ij}s_{ij}\right\} \tag{1.8}$$

S.a:

$$f_i + \sum_{\substack{k \\ k\neq i}} S_{ki} - \sum_{\substack{j \\ j\neq i}} S_{ij} = 0 \quad \forall i \in V \tag{1.9}$$

$$-\overline{S}_{ij} \le S_{ij} \le \overline{S}_{ij} \quad \forall (i,j) \in A \tag{1.10}$$

$$S_{ij} \in \mathbb{R} \tag{1.11}$$

Dónde:

r_{ij} : Costo por unidad de flujo

S_{ij} : Flujo de i a j

f_i : Inyección (nodo fuente, +) o eyección (nodo sumidero, -) de flujo en el nodo i

$\bar{S}_{ij}$: Límite en el flujo de i a j

De manera similar el VRP puede considerarse como un problema de flujo en red, donde el nodo fuente corresponde al depósito y los nodos sumideros son los clientes, cuya demanda de unidades de mercancía corresponde a la eyección de flujo. En este caso la capacidad de flujo corresponde a la capacidad de cada vehículo, también se debe resaltar que el flujo por la red (ruta) es entero, ya que se habla de unidades de mercancía, esto no implica que las variables de flujo sean enteras, pero las variables x_{ij} del problema formulado en (1.12)-(1.22) están directamente relacionadas con las variables de flujo. Esto implica que dichas variables de flujo pueden estar expresadas en función de las variables de los arcos de la ruta del VRP (1.19).

La desigualdad presentada en (1.19) representa la relación de S_{ij} con x_{ij}, si dicho arco está activo hay flujo por dicho arco ($0 \le S_{ij} \le C$) en caso de que x_{ij} sea cero entonces no hay dicho arco y por lo tanto, tampoco hay flujo ($0 \le S_{ij} \le 0 \leftrightarrow S_{ij} = 0$). Adicionalmente el sentido del flujo lo define el arco activo, es decir, si x_{ij} está activo el flujo va de i a j ($S_{ij} \ge 0, S_{ji} = 0$), si es x_{ji} el arco activo, entonces el flujo va de j a i ($S_{ji} \ge 0, S_{ij} = 0$).Finalmente los valores de las variables de flujo, a pesar de ser definidos como tipo real, los resultados obtenidos serán del tipo entero.

Para esta formulación no hay un costo asociado al flujo, las ecuaciones de flujo en red describen el comportamiento de flujo de unidades de mercancía y son discriminadas por depósito (1.17) y por clientes (1.18). Finalmente el modelo de VRP puede ser expresado como:

$$z = \min\left\{ \sum_{i} \sum_{\substack{j \\ i \neq j}} c_{ij} x_{ij} \right\} \qquad (1.12)$$

s.a:

$$\sum_{\substack{i \\ i \neq j}} x_{ij} = 1 \quad \forall j \in V \setminus \{0\} \qquad (1.13)$$

$$\sum_{\substack{j \\ j \neq i}} x_{ij} = 1 \quad \forall i \in V \setminus \{0\} \qquad (1.14)$$

$$\sum_{\substack{j \\ j\neq 0}} x_{0j} = K \tag{1.15}$$

$$\sum_{\substack{i \\ i\neq 0}} x_{i0} = K \tag{1.16}$$

$$\sum_{\substack{j \\ j\neq 0}} S_{0j} = \sum_{\substack{i \\ i\neq 0}} d_i \tag{1.17}$$

$$\sum_{\substack{k \\ k\neq i}} S_{ki} - \sum_{\substack{j \\ j\neq i}} S_{ij} = d_i \quad \forall i \in V \tag{1.18}$$

$$0 \le S_{ij} \le C \cdot x_{ij} \quad \forall (i,j) \in A \tag{1.19}$$

$$S_{i0} = 0 \quad \forall i \in V \setminus \{0\} \tag{1.20}$$

$$S_{ij} \in \mathbb{R} \tag{1.21}$$

$$x_{ij} \in \{0,1\} \tag{1.22}$$

Con la anterior formulación se garantiza que no se generan sub-tour ya que estás no satisfacen las restricciones de flujo en red, tal como se ilustra en el ejemplo de la Figura 1.6.

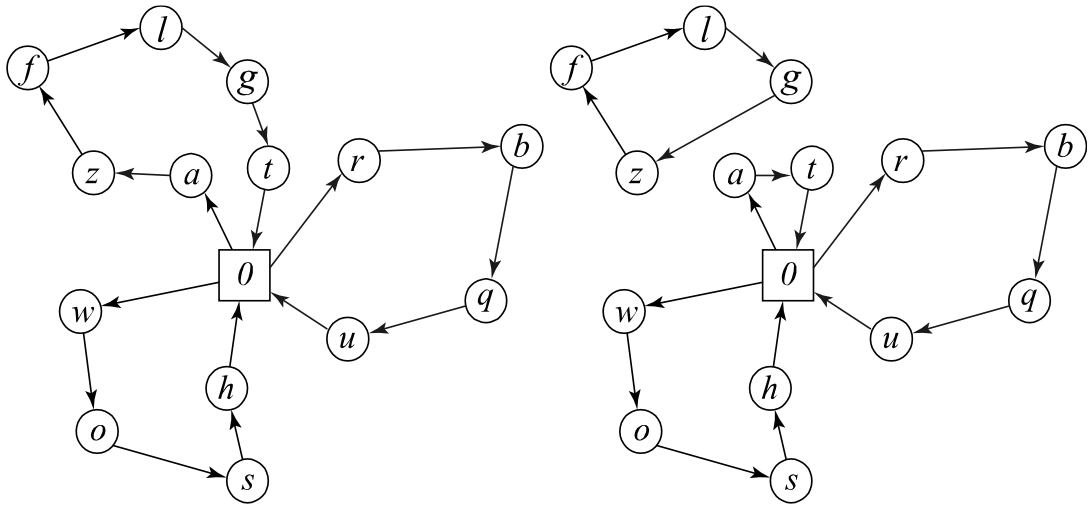


Figura 1.6. (a) Ruta factible, (b) ruta con sub-tour.

En la Figura 1.6 (a) el flujo S_{0a} (proporcionado por el nodo fuente o depósito) es igual a la demanda de los vértices de la ruta, cada vértice extrae la cantidad que demanda, finalmente el flujo por el arco S_{gt} es igual a la demanda en d_t. La Figura 1.6 (b) claramente no satisface las ecuaciones de flujo ya que no hay un nodo fuente que le proporcione la cantidad demandada por los nodos sumideros.

1.4.3 Modelo de Set-Partitioning

Según [1], este modelo es propuesto por [23]. El modelo expuesto en este capítulo se basa en la recopilación de los autores de [1], algunos cambios de notación son empleados con el fin de no generar confusión con símbolos ya empleados.

El modelo de Set-Partitioning, parte de un conjunto de *q* rutas factibles pertenecientes al conjunto *H*, estas rutas son posibles soluciones para un solo vehículo. Donde y_j es una variable binaria que representa el estado de la ruta *j* en la solución. Si esta está en uno significa que la ruta *j* hace parte de la solución óptima. Dicha ruta tiene un costo asociado c_j y unos nodos asociados, este último es representado mediante el parámetro a_{ij}, el cual es de carácter binario. Si está en 1, indica que el vértice *i* hace parte de la ruta *j*.

El modelo matemático que describe el problema del CVRP mediante el uso de las definiciones anteriormente dadas está descrito por las expresiones (1.23) al (1.26). Optimizar el problema consiste en encontrar la combinación adecuada rutas (pertenecientes a *H*) que generen el mínimo costo. Tener todas las posibles combinaciones de rutas generadas haría el problema complejo, en términos de la cantidad de variables, por tal motivo se recurre a un conjunto de rutas diminuto en comparación con lo anteriormente dicho.

$$\min_{y}\left\{\sum_{j\in H} c_j y_j\right\} \tag{1.23}$$

s.a:

$$\sum_{j=1}^{q} a_{ij} y_j = 1 \quad \forall i \in V \setminus \{0\} \tag{1.24}$$

$$\sum_{j\in H} y_j = K \tag{1.25}$$

$$y_j \in \{0,1\} \quad \forall j \in H \tag{1.26}$$

La solución obtenida es óptima en términos del conjunto de rutas empleado, por lo tanto dicha solución no es necesariamente el óptimo global del problema. En este caso las rutas son generadas por la metodología descrita en la sección 3.8. El cual corresponde a un algoritmo colonia de hormigas, con algunas modificaciones. Dicho algoritmo entrega un conjunto de rutas, denominado lista élite. Esta lista élite corresponde al conjunto *H*. Adicionalmente el modelo de Set-Partitioninig requiere de

una técnica de solución exacta. En este caso se recurre a un método de arborescencia denominado B&B. Para dicha metodología es necesario recurrir al capítulo 4 y 5 de este trabajo.

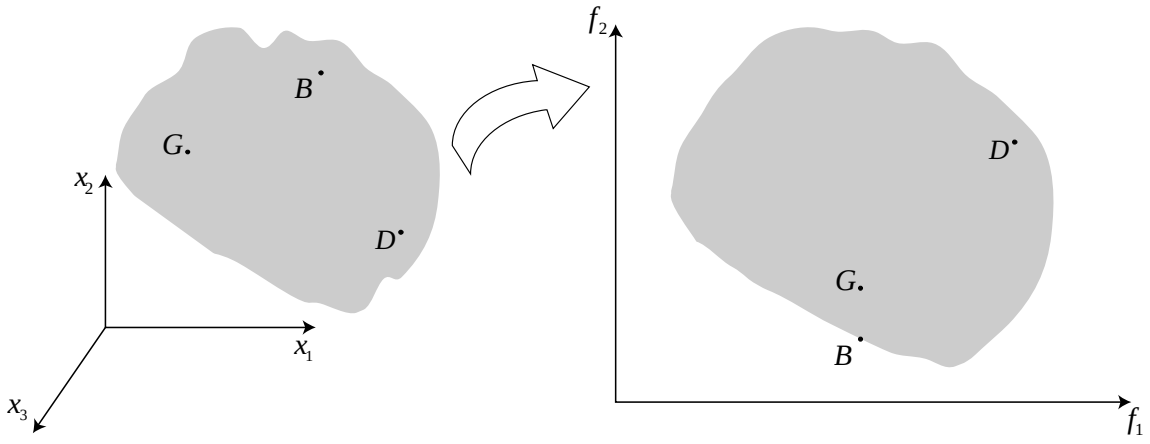
2 Método de reducción del espacio de búsqueda

2.1 Introducción

La disminución del espacio de búsqueda conduce a una reducción del tiempo de cómputo en los problemas de optimización, este consiste en un subconjunto del espacio de soluciones cuyo tamaño es más reducido. Se debe ser cauto con la creación de este subconjunto, ya que es posible que el óptimo del problema quede por fuera de este subconjunto. En esta sección se presenta un método que reduce el espacio de búsqueda basado en el concepto de dominancia empleado en la programación multiobjetivo. Se aclara que no se trata de realizar un problema multiobjetivo, pero las cualidades de dicho concepto y algoritmo permiten implementar una metodología que saca una lista de arcos candidatos sobre los cuales se construyen soluciones al problema del CVRP.

2.2 Concepto de dominancia

El concepto de dominancia es empleado para comparar dos resultados entre sí respecto a un conjunto de M objetivos, para este caso no existe una solución única, sino un conjunto de soluciones, el cual corresponde al frente óptimo de Pareto o frente no dominado. El frente no dominado corresponde al conjunto de soluciones de mejor calidad respecto a un conjunto de puntos del espacio de soluciones (los puntos del frente dominan a dicho conjunto de puntos). Sin embargo las soluciones de un mismo frente no pueden clasificarse como mejores o peores entre sí (ninguna domina a la otra).



(a) *(b)*

Figura 2.1 Espacio de soluciones(a) y espacio objetivo (b).

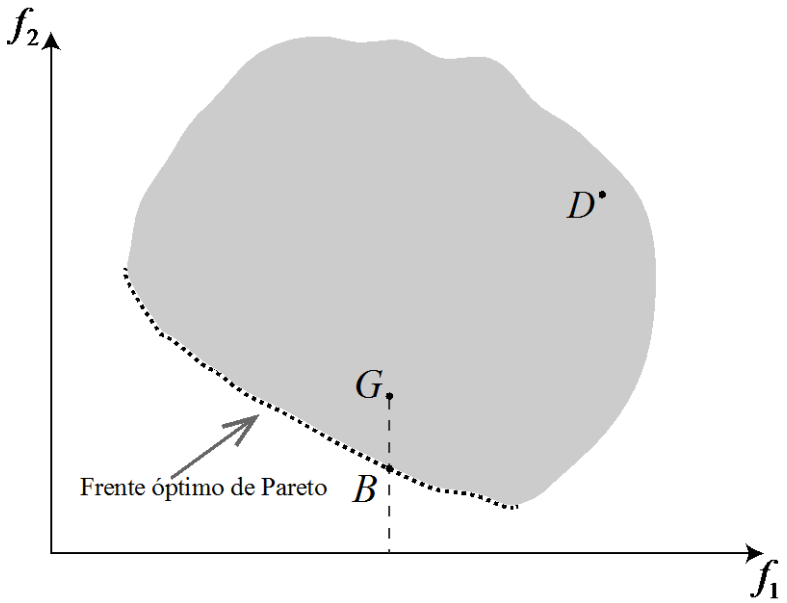


Figura 2.2 Frente no dominado en el espacio objetivo.

En la Figura 2.1 se ilustra un problema de optimización de 3 variables y 2 objetivos. La Figura 2.1 (a) corresponde al espacio de búsqueda, la Figura 2.1 (b) corresponde al espacio objetivo. Existe una correspondencia entre ambos espacios, es decir para el espacio de funciones objetivos, cada par de puntos (f_1, f_2) corresponde a un valor de (x_1, x_2, x_3) y viceversa. El frente no dominado es obtenido del espacio de objetivos tal como se muestra en la Figura 2.2, dicho frente posee mejores soluciones (f_1, f_2) respecto a los demás puntos (como por ejemplo el punto D). Para este ejemplo el frente resaltado se denomina frente óptimo de Pareto, otros frentes pueden constituirse pero son dominados por el frente óptimo de Pareto.

Basado en la referencia [24], se introduce la notación $f_i\left(X^{(a)}\right) \lhd f_i\left(X^{(b)}\right)$, para indicar que la función objetivo f_i evaluado en el punto $X^{(a)}$ es mejor a la función objetivo f_i evaluado en el punto $X^{(b)}$. De manera similar $f_i\left(X^{(b)}\right) \rhd f_i\left(X^{(a)}\right)$ indica que la función objetivo f_i evaluado en el punto $X^{(b)}$ es peor que la función objetivo f_i evaluado en el punto $X^{(a)}$.

De acuerdo a lo anterior, se dice que $X^{(a)}$ domina a $X^{(b)}$ $\left(X^{(a)} \preceq X^{(b)}\right)$ si se cumple los 2 siguientes criterios:

1. La solución $X^{(a)}$ no es peor que $X^{(b)}$ en todos los objetivos $\left(f_i\left(X^{(a)}\right) \not\trianglerighteq f_i\left(X^{(b)}\right) \quad \forall i = 1, \cdots, M\right)$.
2. La solución $X^{(a)}$ mejor que $X^{(b)}$ por lo menos en un objetivo $\left(f_i\left(X^{(a)}\right) \trianglelefteq f_i\left(X^{(b)}\right) \quad \forall i \in \{1, \cdots, M\}\right)$.

De acuerdo al ejemplo de la Figura 2.2, el concepto de dominancia para el punto B consiste en no ser peor en todos los objetivos evaluados, respecto a otro punto en comparación G (D), adicionalmente, si dicho punto B es mejor por lo menos en uno de sus objetivos entonces se dice que el punto G (D) es dominado por B y por lo tanto el punto G (D) no pertenece al frente no dominado.

Los puntos G y D no constituyen el frente óptimo de Pareto pero pueden hacer parte de otros frentes los cuales dominan ciertos puntos del espacio de soluciones, en la sección posterior se indica la forma para construir frentes no dominados basados en el concepto de dominancia.

2.3 Construcción de Frentes no Dominados

En la sección anterior se muestra un ejemplo de minimización para las 2 funciones objetivos. Pero hay *M* combinado 2 de posibilidades. Para el caso de 2 objetivos (tal como se ilustra en la Figura 2.3), estas son:

- La función objetivo 1 y 2 son de minimización (Figura 2.3 *(a)*).
- La función objetivo 1 es de minimización y la 2 es de maximización (Figura 2.3 *(b)*).
- La función objetivo 1 es de maximización y la 2 es de minimización (Figura 2.3 *(c)*).
- La función objetivo 1 y 2 son de maximización (Figura 2.3 *(d)*).

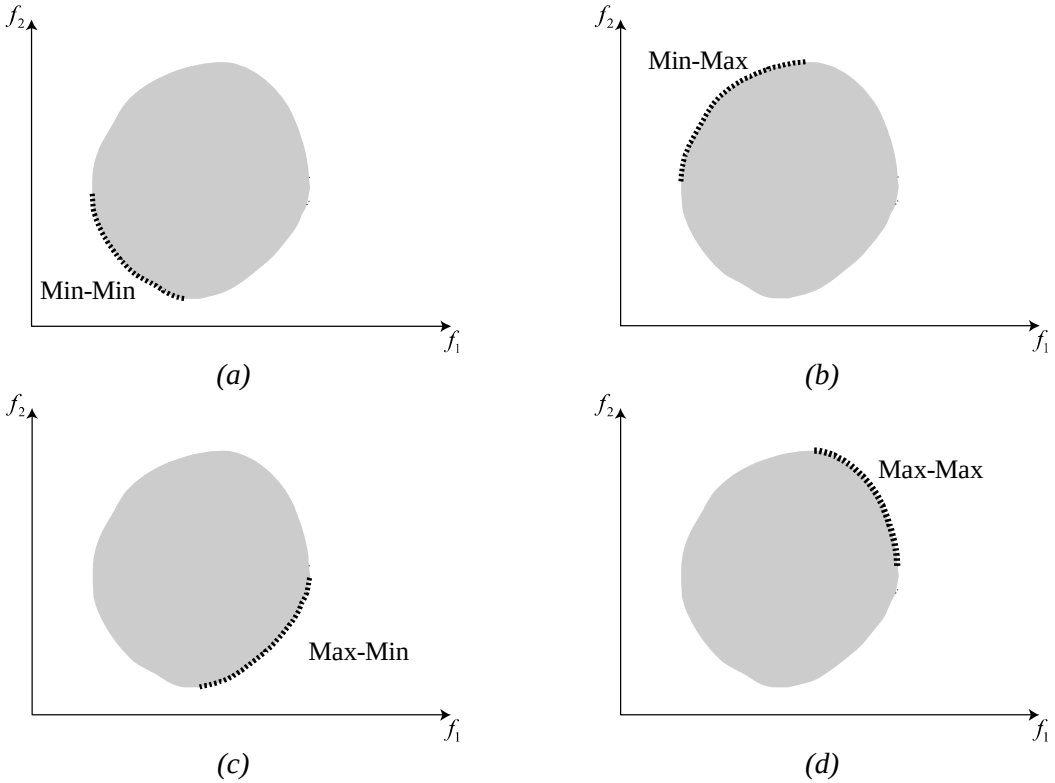


Figura 2.3 Frentes óptimos de Pareto para un problema de 2 objetivos y las variantes en el tipo de problema.

En el espacio de soluciones podrían construirse infinitos frentes no dominados. Para las técnicas metaheurísticas se habla de espacio de búsqueda que es un subconjunto del espacio de soluciones, esto permite obtener una cantidad finita de frentes no dominados.

Partiendo de un problema min-min y 2 objetivos, el proceso para obtener frentes es el siguiente.

1. *Partir de un conjunto de puntos* F_i*, en un principio contiene todos los puntos del espacio de búsqueda*
2. *Elegir un punto* $X^{(a)}$ *de* F_i *y eliminar los puntos de* $X^{(j)}$ *(que también pertenecen a* F_i*) que son dominados por este.*
3. *Si toda la lista de puntos* F_i *actual ha sido seleccionada ir al paso 4, de lo contrario retorne al paso 2.*

4. *Si en las etapas anteriores no se eliminaron puntos, entonces el* F_i *corresponde al último frente y se debe parar. De lo contrario el conjunto de puntos almacenados hasta el momento en* F_i *corresponde al frente no dominado i.*
5. *Se debe actualizar i, i=i+1. Este nuevo* F_i *estará conformado por los puntos eliminados, que no pertenezcan a los anteriores frentes y se debe regresar al paso 2.*

De lo anterior se resalta que cuando *i*=1 se obtiene el frente óptimo de Pareto del espacio de búsqueda F_1 (Distinto al frente óptimo de Pareto del espacio de soluciones). Los demás frentes son el conjunto de frentes no dominados.

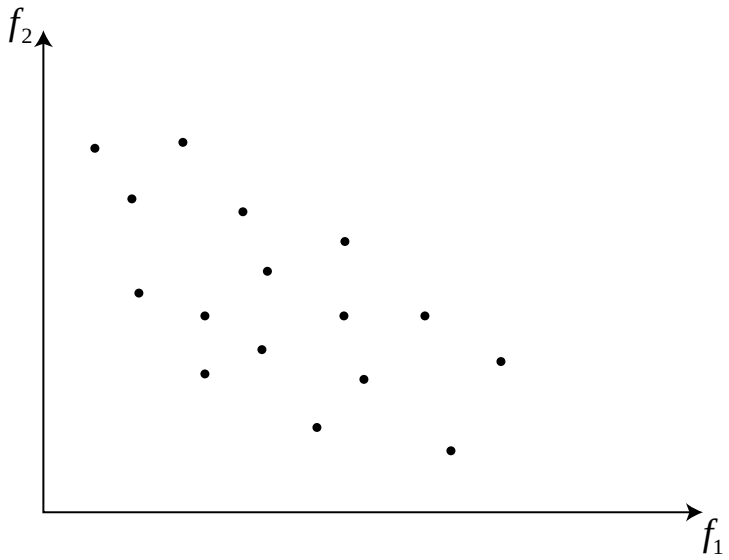
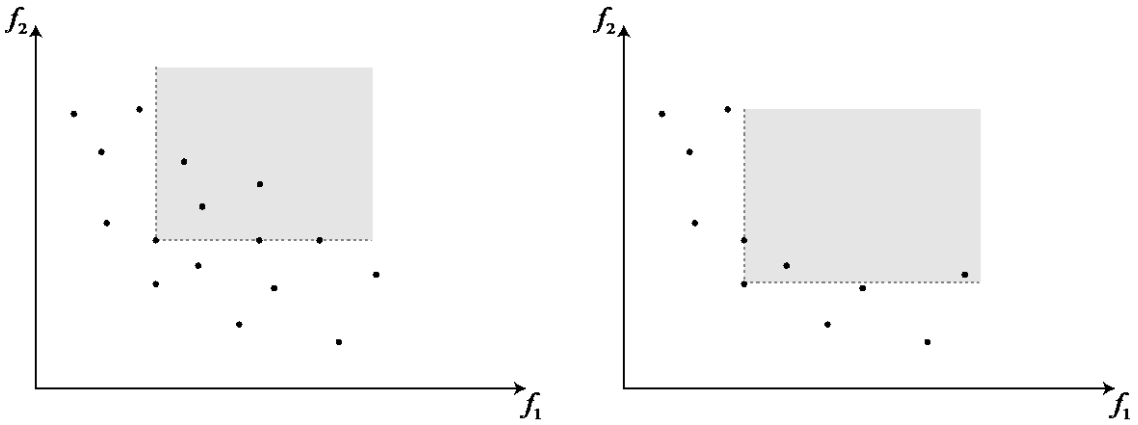


Figura 2.4 Ejemplo para la construcción de frentes no dominados.

Para dar mayor claridad a lo descrito se recurre al ejemplo plantado en la Figura 2.4. Se crea un conjunto de puntos, denominado F_1 el cual contiene en un principio todos los puntos del problema. Partiendo de dicho conjunto, se selecciona un punto cualquiera y se observa cuales puntos son dominados por este, tal como se muestra en Figura 2.5(a) (aquellos que están en el rectángulo gris), dichos puntos dominados son eliminados de F_1.

El proceso de eliminar aquellos puntos dominados se repite tal como se ilustra en Figura 2.5, hasta que se obtiene un conjunto de puntos que no son dominados, en este caso este corresponde al frente óptimo de Pareto del espacio de búsqueda (Figura 2.6).



(a) *(b)*

Figura 2.5 Eliminación de los Puntos dominados

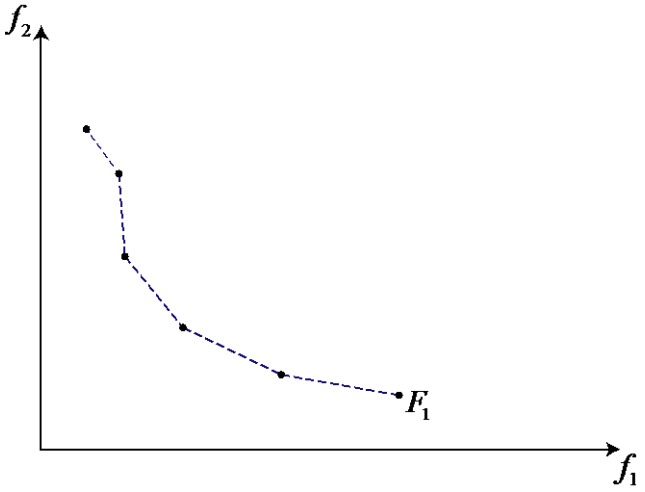


Figura 2.6 Frente óptimo de Pareto.

Posteriormente se crea un nuevo conjunto F_2, el cual contiene los puntos eliminados en la etapa previa. De nuevo el proceso se repite, sin tener en cuenta los puntos que ya hacen parte de un frente, Este ciclo finaliza cuando en la lista de puntos actuales (el último conjunto) no se eliminan más puntos, finalmente se obtiene el conjunto de frentes ilustrado en la Figura 2.7.

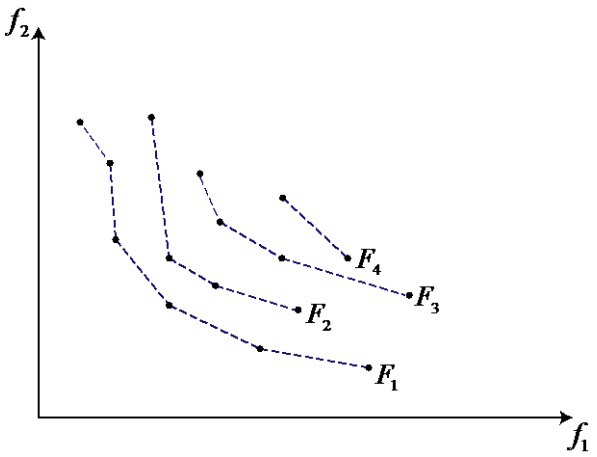


Figura 2.7 Conjunto de frentes no dominados.

Las métricas descritas para obtener cada frente son intuitivas, sin embargo no es la manera más eficiente para obtenerlos, ya que esto implica que cada punto seleccionado debe compararse con los restantes para saber cuáles son dominados por este. Existe un algoritmo más eficiente, propuesto por Kung [25] para obtener los frentes no dominados para problemas de *M* objetivos, sin embargo, para este caso se recurre al uso de 2 objetivos, las razones son expuestas en las subsecciones posteriores.

2.4 Método de Kung para la construcción de frentes no dominados

Este método es propuesto en [25], las bases teóricas son tomadas de la referencia [26]. El algoritmo parte de la lista de soluciones o individuos ordenados de acuerdo a uno de sus objetivos, se ordena del mejor al peor objetivo, si es problema de minimización el mejor es el mínimo, si es un problema de maximización el mejor es el máximo. Una vez ordenados se hace una partición de la lista obtenida en 2 listas denominadas izquierda y derecha, siendo la lista de la izquierda mejor respecto a la función objetivo 1, esta partición es recursiva hasta obtener la lista Derecha e Izquierda con un solo individuo cada uno, a partir de estas 2 últimas listas se analiza el concepto de dominancia expuesto en la sección 2.2. Para el resto de objetivos (ya que Izquierda es mejor que la derecha en f_1). Si uno de los individuos es dominado por el otro entonces este es eliminado de todas las listas generadas recursivamente. El proceso regresa de la lista con menor cantidad de individuos hasta la lista original, comparando si el último de la lista de Izquierda domina al último de la lista de Derecha (ya que estos tienen mejor f_2 dentro de su propio conjunto), ya que si esto es válido este individuo de la lista Izquierda domina todos los otros individuos de la lista Derecha. Lo contrario implica que es posible

que hay individuos no dominados en la lista Derecha. Este algoritmo recurre de una etapa 2, para comprar los individuos del conjunto resultante, ya que en la 1 se compararon entre los mejores de cada conjunto, es posible que los individuos de un mismo conjunto generado domine al otro, por lo que se hace necesario dicha etapa.

El ejemplo ilustrativo de la Figura 2.8, permite explicar dicha metodología, los datos son mostrados en la Tabla 2.1. Este consiste en un problema de 2 objetivos, ambos de minimización. Se debe partir del conjunto de los índices de los individuos o índices de las soluciones exploradas ordenadas por el objetivo uno (f_1) del mejor al peor valor, en este caso como el objetivo 1 es de minimización entonces se ordena de manera ascendente (Figura 2.9).

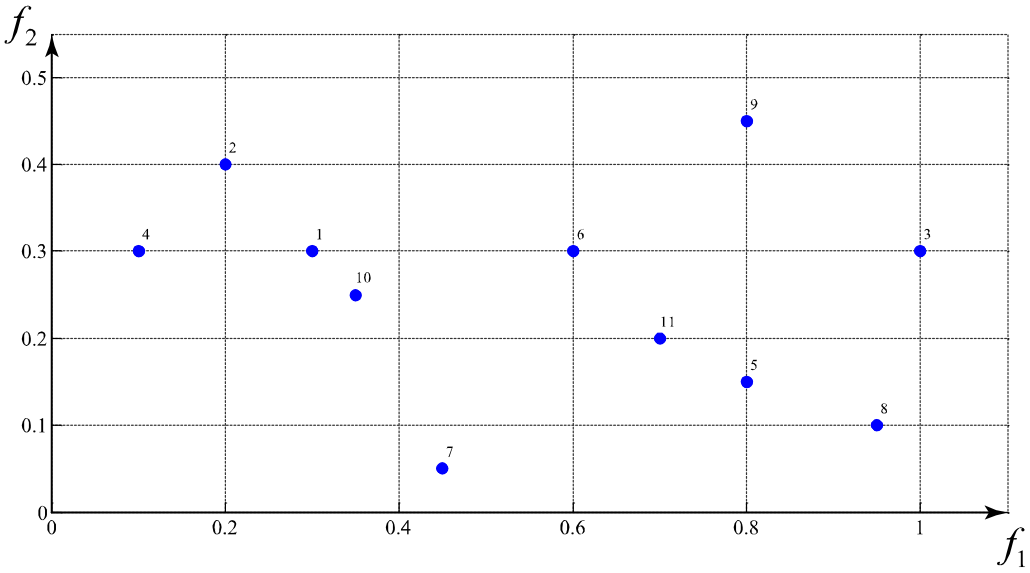


Figura 2.8. Ejemplo para la aplicación del método de Kung,

Individuo	f_1	f_2	*Individuo*	f_1	f_2
1	0,3	0,3	7	0,45	0,05
2	0,2	0,4	8	0,95	0,1
3	1	0,3	9	0,8	0,45
4	0,1	0,3	10	0,35	0,25
5	0,8	0,15	11	0,7	0,2
6	0,6	0,3			

Tabla 2.1. Tabla de datos del ejemplo de la Figura 2.8.

4	2	1	10	7	6	11	5	9	8	3

Figura 2.9. Ejemplo de frentes no dominados mediante Kung.

Siendo M conjunto actual de *n* individuos, la posición a partir dicho conjunto está dada por la expresión (2.1).

$$n_{izq} = \left\lceil \frac{n}{2} \right\rceil \tag{2.1}$$

$$n_{der} = n - n_{izq} \tag{2.2}$$

El conjunto *M* es divido en 2 conjuntos, M^{izq} con n_{izq} individuos (2.1) y M^{der} con n_{der} individuos (2.2). Este proceso es cíclico hasta obtener conjuntos M^{izq} y M^{der} de un solo individuo. Adicionalmente a M^{izq} hay asociado un conjunto *I*, el cual es un subconjunto de este ($I \subseteq M^{izq}$). De manera similar M^{der} tiene asociado *D*, el cual es subconjunto de este ($D \subseteq M^{der}$).

Se debe partir la lista de individuos de manera recursiva y almacenara la lista de M^{izq} y M^{der}, generado en cada división (Figura 2.10), en este caso se generaron 4 niveles, debido a la partición sucesiva de conjuntos. En el último nivel se cumple las propiedades descritas en las expresiones

$$I = M^{izq} \tag{2.3}$$

$$D = M^{der} \tag{2.4}$$

El proceso de partir del nivel 0 hasta el último nivel podría denominarse barrido hacia abajo, en dicho barrido se ha generado los todos los subconjuntos M^{izq} y M^{der}. En el barrido hacia arriba se construye los subconjuntos *I* y *D.*

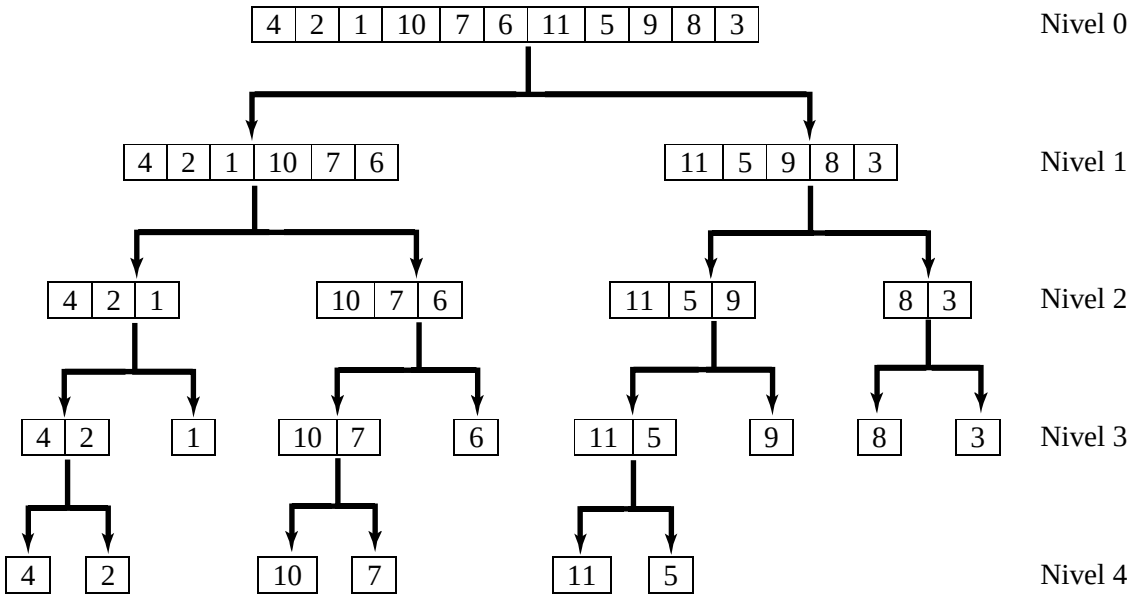


Figura 2.10. Conformación de los subconjuntos M^{izq} y M^{der}.

Partiendo del nivel 4 y siempre por izquierda. El individuo 4 tiene mejor f_2 que la del individuo 2, por lo tanto 4 pasa a formar el conjunto *I* en el Nivel 3. Estando en este nivel, se observa si el último individuo del conjunto *D* (individuo 4 ya que el 2 fue descartado) es mejor al último del conjunto *I* (individuo 1), en este caso *D* no es mejor a *I*, por lo tanto 4 y 1 pasan al Nivel 2 para formar el conjunto de *I* de dicho nivel $(I \cup D)$.

Ya en el nivel 2 es necesario empezar de nuevo en el nivel 4 con los individuos 10 y 7 para poder saber cómo queda el conjunto *D* del nivel 2. En este caso 10 no tiene mejor f_2 que 7, por lo tanto estos se unen para conformar *I* en el Nivel 3. Ya en el nivel 3 se observa si el último elemento del conjunto *I* (individuo 7) tiene mejor f_2 que el último individuo del conjunto *D* (individuo 6). En este caso 7 tiene mejor f_2 que 6 y este último no formará parte del conjunto *D* del Nivel 2 (Figura 2.12).

En el nivel 2 se observa si el último individuo del conjunto *I* (individuo 1) tiene mejor f_2 que el último del conjunto *D* (individuo 7 ya que 6 fue descartado), en este caso 1 no es mejor que 7, entonces el conjunto *I* del Nivel 1 está conformado por la unión de *I* y *D* del Nivel 2 (Figura 2.12).

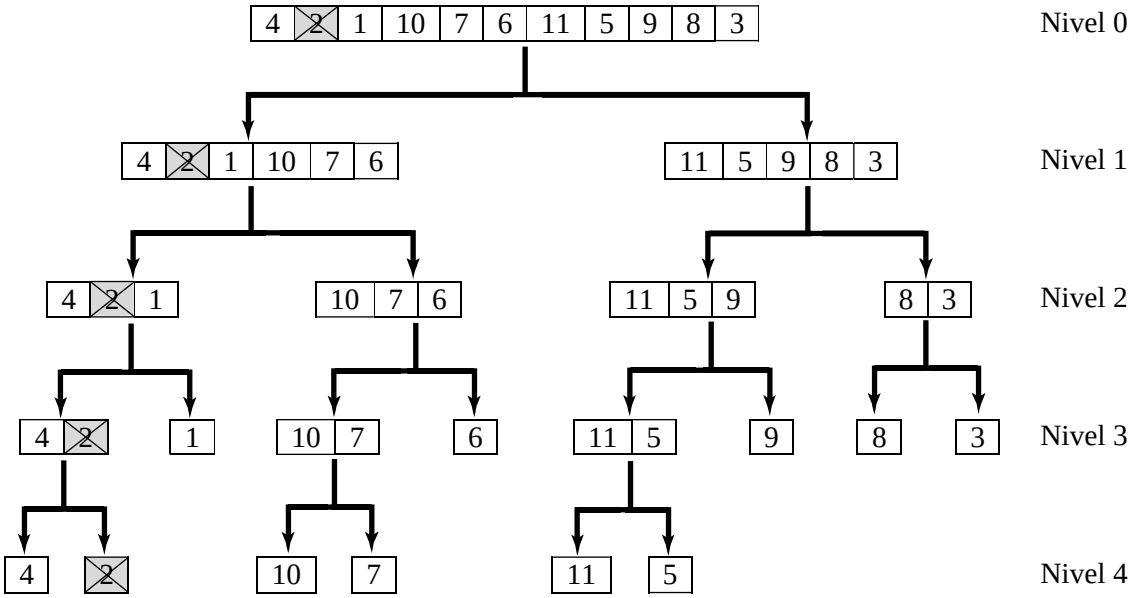


Figura 2.11. Conformación de los subconjuntos I y D, parte 1.

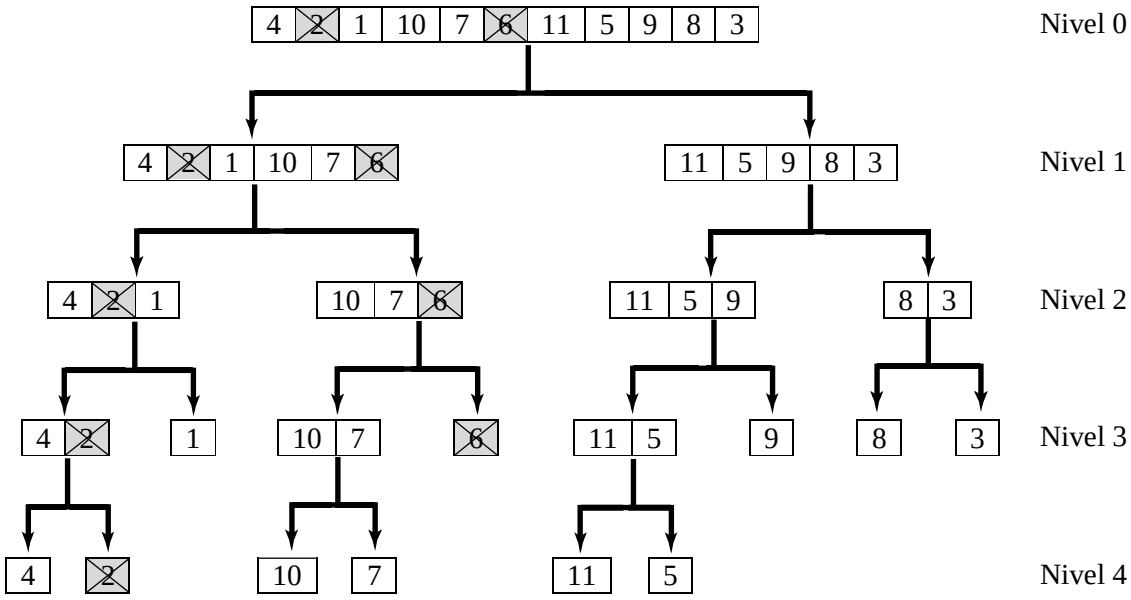


Figura 2.12. Conformación de los subconjuntos I y D, parte 2.

Una vez en el nivel 1 es necesario saber cómo queda los conjuntos de la derecha de dicho nivel. Se realiza el mismo procedimiento aplicado hasta al momento, la Figura 2.13 representa como quedan los conjuntos *I* y *D* del Nivel 1.

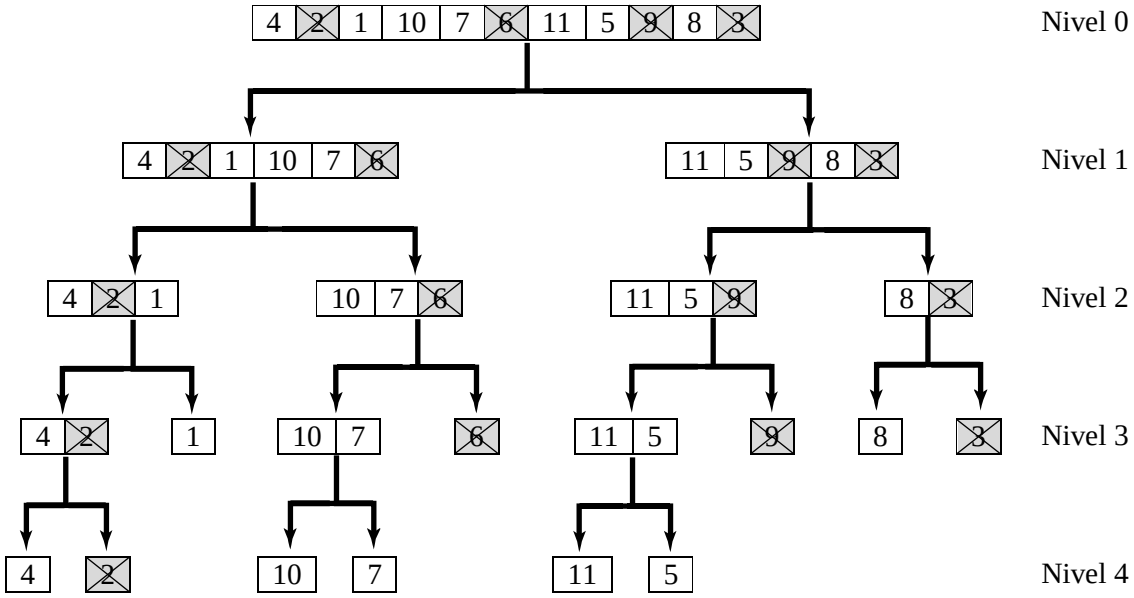


Figura 2.13. Conformación de los subconjuntos I y D, parte 2.

Finalmente en el nivel uno, los conjuntos están conformados de la siguiente manera:

$$I = \{4,1,10,7\}$$
$$D = \{11,5,8\}$$

En este caso el último individuo de *I* (4) es mejor (respecto a f_2) que el último individuo de *D* (8). La Figura 2.14 muestra los individuos no descartados en el nivel, estos son los candidatos a conformar el primer frente no dominado (conjunto *F*), sin embargo es posible que individuos de un mismo conjunto *D* (o *I*) sean dominados por uno o varios de estos, ya que sobre el mismo conjunto *D* (o *I*) no se realizaron las comparaciones.

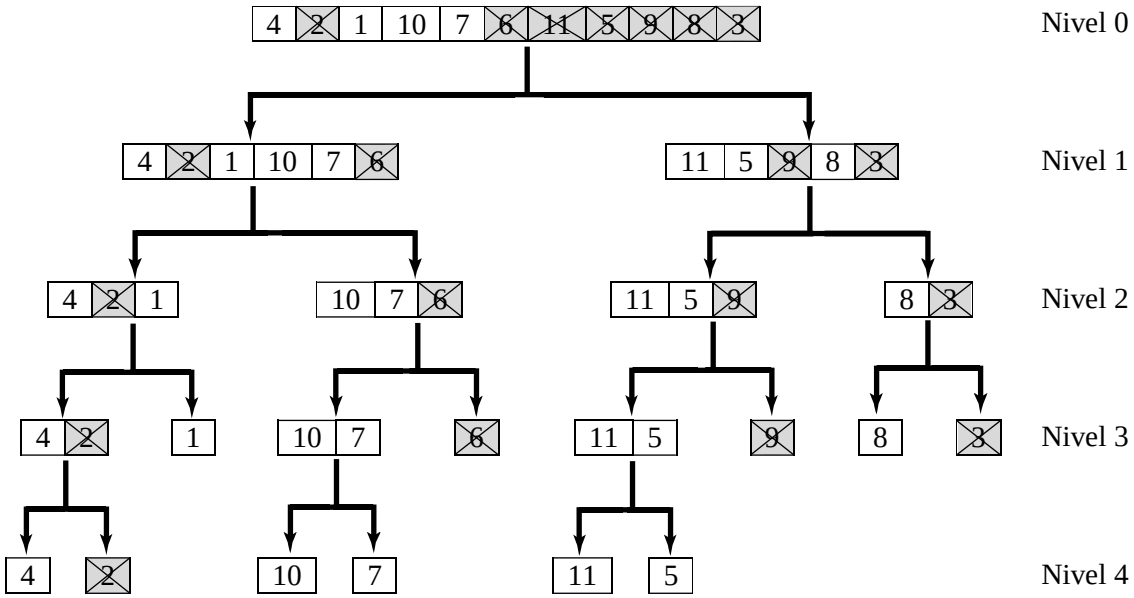


Figura 2.14. Conformación de los subconjuntos I y D, parte 3.

$$F = \{4,1,10,7\}$$

Siendo F_i el elemento *i* del conjunto *F*. La etapa 2 consiste en verificar de izquierda a derecha si F_i es mejor (respecto a f_2) a F_{i+1}, en caso de ser mejor, se elimina el individuo F_{i+1}, en caso contrario, se actualiza *i=i+1* y se repite la comparación.

Para este caso, el individuo 4 es mejor que el individuo 1, por lo tanto este es eliminado:

$$F = \{4,10,7\}$$

4 no es mejor que 10, por lo tanto se procede con la siguiente comparación, donde se comprueba que 10 no es mejor que 7. Finalmente el primer frente está conformado por los individuos 4, 10 y 7. El procedimiento empieza nuevamente con los individuos que no forman parte de F, estos son 2, 1,6 11, 5, 9, 8 y 3. El algoritmo finaliza cuando no haya más individuos para conformar nuevos frentes. La Figura 2.15, ilustra cómo quedan conformados los frentes.

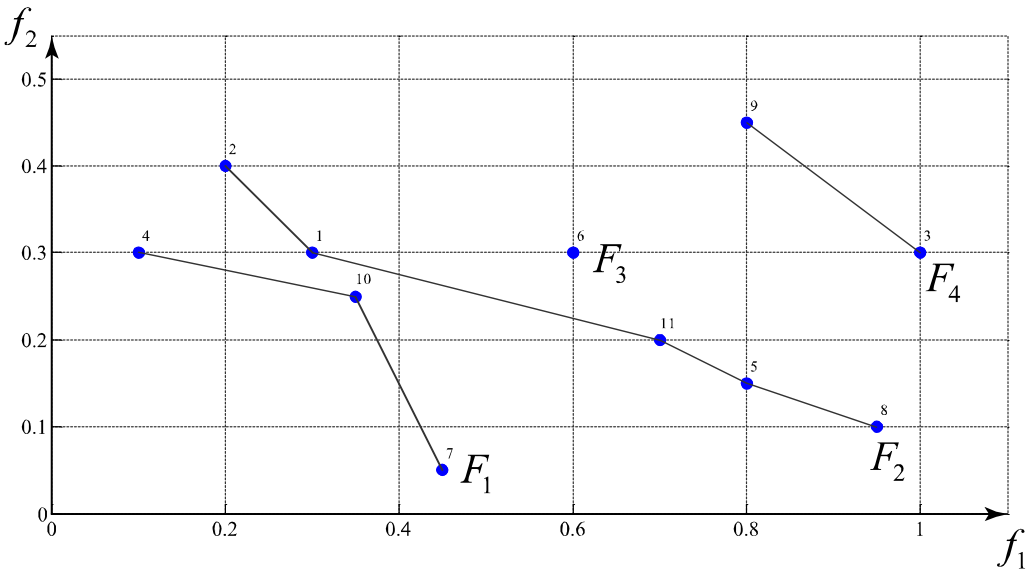


Figura 2.15. Conformación de los frentes no dominados, mediante el método de Kung.

2.4.1 Algoritmo para la obtención de un frente no dominado

En esta sub-sección se presenta el Pseudocódigo para obtener un frente, el algoritmo se basa en la recursividad para la división sucesiva de conjuntos, dividiendo primero siempre por el lado izquierdo.

Etapa1:

- Dato de entrada *M*.
- Dato de salida *F*.

Variables Globales:

$f2EsMin$: Bandera binaria, verdadero si la función objetivo 2 (f_2) es de minimización

f_2 : Vector de la función objetivo 2 para todos los individuos, organizadas según el índice del individuo

Variables locales:

M: Vector actual de Individuos actuales ordenados por el objetivo 1

M^{izq} : Parte izquierda del vector M

M^{der} : Parte derecha del vector M

F: Vector resultante según la dominancia

D: Vector actual de Derecha

I: Vector actual de Izquierda

i: Contador de los índices

Los vectores mencionados almacenan los índices de los individuos según el caso

Función→ *F=**OrdenFrentes** (M)*
Inicio:

Si $n \leq 1$***, entonces***
Inicio:

$n_{izq} = \left\lceil \frac{n}{2} \right\rceil ;$

$n_{der} = n - n_{izq} ;$

Para $i = 0, \cdots, n$
Inicio:

Si $i \leq n_{izq}$
Inicio:

$M_i^{izq} = M_i ;$

Fin
De lo contrario
Inicio:

$M_i^{der} = M_i ;$

Fin;
Fin;
*I=**OrdenFrentes** (M^{izq}); /* Salta a dividir por izquierda*/*
*D=**OrdenFrentes** (M^{der}); /* Salta a dividir por derecha*/*
/ Inicio de la comparación de sub-conjuntos*/*

Si $\left\{ \begin{array}{c} \left[\left((f_2)_{D_{n_{der}}} \leq (f_2)_{I_{n_{izq}}} \right) \wedge \left(f2EsMin \right) \right] \\ \vee \\ \left[\left((f_2)_{D_{n_{der}}} \geq (f_2)_{I_{n_{izq}}} \right) \wedge \left(no\left(f2EsMin \right) \right) \right] \end{array} \right\}$***, entonces***

Inicio:

$F = M^{izq} \cup M^{der} ;$

Fin
De lo contrario
Inicio:

$F = M^{izq} ;$

Fin;
Fin
De lo contrario
Inicio:

$F = M ;$

Fin;
Fin;

Etapa 2:

- Dato de entrada *F*.
- Dato de salida *F*.

Variable Global:

$f2EsMin$: Bandera binaria, verdadero si la función objetivo 2 (f_2) es de minimización

f_2 : Vector de la función objetivo 2 para todos los individuos, organizadas según el índice del individuo

Variables locales:

F: Vector resultante según la dominancia
i: Contador de los índices
m: Tamaño de F

Función→ *F=**Fase2OrdenFrentes** (F)*
Inicio:
$i = 1$
m= Tamaño de F;
Si $m > 1$***, entonces***
Inicio:
Mientras *Verdadero* ***, hacer***
Inicio:

$$\textbf{Mientras} \left\{ \begin{array}{c} \left[\left((f_2)_{F_i} \le (f_2)_{F_{i+1}} \right) \wedge \left(f2EsMin \right) \right] \\ \vee \\ \left[\left((f_2)_{F_i} \ge (f_2)_{F_{i+1}} \right) \wedge \left(no\left(f2EsMin \right) \right) \right] \end{array} \right\} \textbf{, hacer}$$

Inicio
EliminarPosicion*(i,F); /*Elimina la posición i de F*/*
m= Tamaño de F;
Si $m \le i$ ***, entonces***
Inicio:
Break;
Fin;
Fin;
$i = i + 1$*;*
Si $i \ge m$***, entonces***
Inicio:
Break;
Fin;
Fin;
Fin;

La función ***EliminarPosicion****(i,F)* elimina la posición *i* del vector *F* y reajusta el tamaño (las dimensiones) de *F* para que no queden vacíos.

2.4.2 Algoritmo para obtener todos los frentes no dominados

Partiendo de las siguientes variables:

- *Individuo*: almacena los índices individuos en el orden ascendente de dicho índice
- f_1: Vector de la función objetivo 1 para todos los individuos, organizadas según el vector *individuo*
- f_2: Vector de la función objetivo 2 para todos los individuos, organizadas según el vector *individuo*
- $f1EsMin$: Bandera binaria, verdadero si la función objetivo 1(f_1) es de minimización
- $f2EsMin$: Bandera binaria, verdadero si la función objetivo 2 (f_2) es de minimización
- *R*: Número actual de individuos que aún no se les ha asignado un frente
- *n*: Tamaño del vector de individuos (número total de individuos)
- *VF* : Arreglo de Arreglos, este es una lista de los frentes conformados. No es una matriz como tal, ya que las dimensiones en columnas varía según la cantidad de individuos de un frente. VF_i hace alusión al frente i, en este está almacenado los individuos de dicho frente (Figura 2.16).

	VF				
VF_1	4	10	7		
VF_2	2	1	11	5	8
VF_3	6				
VF_4	9	3			

Figura 2.16. Frentes no dominados (VF) para el ejemplo de la Figura 2.8.

Se tiene que el diagrama de flujo de la Figura 2.17 corresponde al procedimiento realizado para clasificar los individuos en los frentes no dominados.

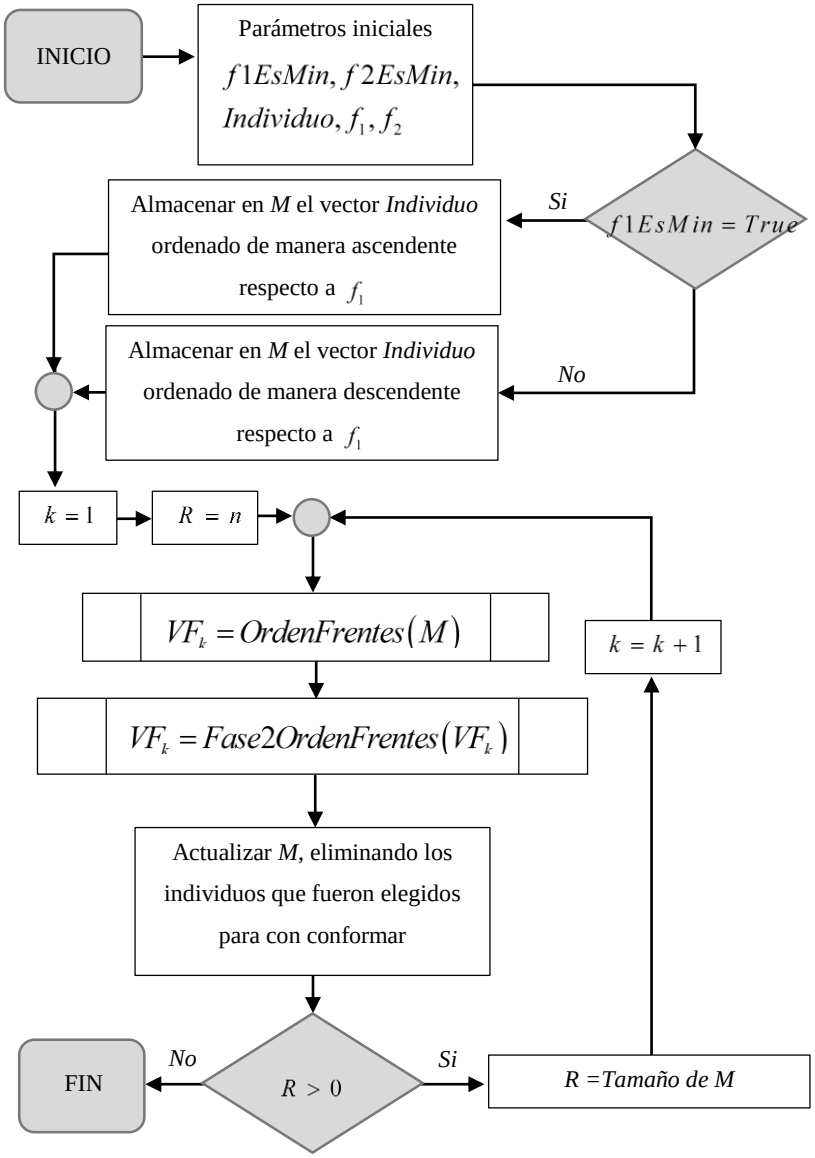


Figura 2.17. Diagrama de flujo del método de Kung para obtener los frentes no dominados con 2 funciones objetivo.

2.5 Metodología propuesta para la reducción del espacio de búsqueda

Con la metodología descrita en las secciones anteriores de este capítulo, se implementa el método para la reducción del espacio de búsqueda. Partiendo de un conjunto de clientes y un depósito geográficamente referenciados, suponer que el objetivo 1 (f_1) corresponde a todas las coordenadas en el *x* de todos los vértices del sistema y el objetivo 2 (f_2) corresponde a todas las coordenadas en el eje *y*.

Al aplicar la metodología de frentes se observa que no solo se generan frentes no dominados en términos de programación multiobjetivo, sino que también se generan arcos hablando en términos del VRP, por lo tanto los arcos obtenidos será la reducción del espacio de búsqueda, sin embargo, la creación de este sub-espacio debe ser tal que se puedan generar soluciones factibles del problema. Por tal motivo surge la metodología descrita a continuación, la cual emplea el método de frentes no dominados anteriormente descrito, para así obtener un espacio de búsqueda reducido.

El método consiste en obtener los frentes no dominados en múltiples escenarios, uno de estos escenarios consiste en rotar los clientes alrededor del depósito y recalcular los frentes, la rotación es de 360° en n_p pasos pre especificados. Evaluar múltiples escenarios implica que es posible que un arco se obtenido múltiples veces, la cantidad de veces que un arco es obtenido es almacenado en una matriz de contadores. En este caso el problema es simétrico, por lo tanto el arco *i-j* es igual que el arco *j-i.*

Variables:

- Considerar $\left(x_i, \mathrm{y}_i\right)$ como el par coordenado del cliente *i* y $\left(x_0, \mathrm{y}_0\right)$ como el par coordenado del depósito
- *Nv*: Matriz simétrica asociada a cada arco (de tamaño $n*n$, siendo *n* el número de clientes), cuyas componentes nv_{ij} indica el número de veces que se obtuvo el arco *i-j* o *j-i* en el proceso de frentes no dominados
- Pa : Parámetro de aceptación, un arco *i-j* o *j-i* es aceptado en el espacio de búsqueda reducido, si se cumple (2.5)

$$nv_{ij} \geq Pa\left(n_p\right) \qquad (2.5)$$

- *Mb*: es una matriz simétrica de $\left(n+1\right)*\left(n+1\right)$ cuya componente m_{ij} indica si el arco *i-j* o *j-i* pertenece al espacio de búsqueda reducido (con valor 1 si pertenece y 0 lo contrario), de acuerdo al criterio establecido en (2.5). Adicionalmente todos los arcos que salen y entran al depósito son considerados (2.6)

$$m_{0j} = m_{j0} = 1 \quad \forall j \neq 0 \qquad (2.6)$$

Procedimiento:

1. *Ubicar el depósito como el origen de los ejes coordenados Figura 2.18.*

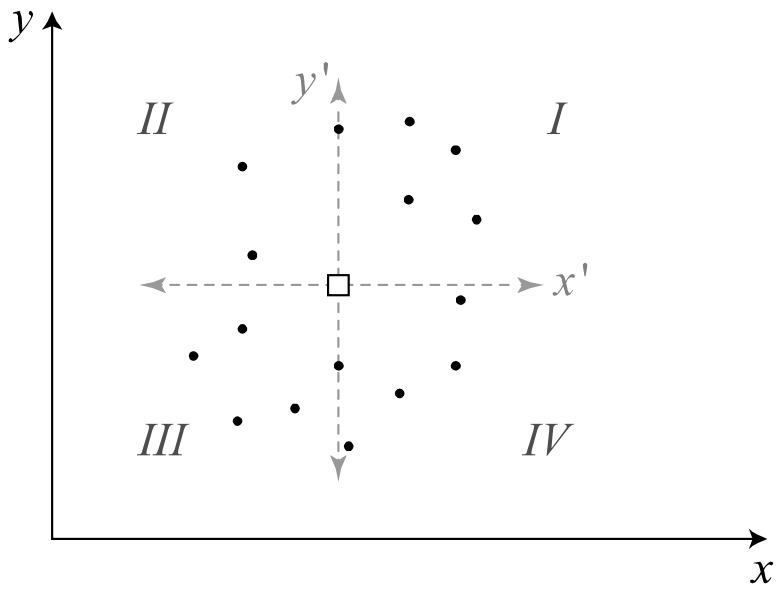


Figura 2.18. División por cuadrantes de los clientes del problema.

2. *Calcular los frentes no dominados para los cuatro posibles casos mencionados en la sección 2.3, estos son:*
    - *La función objetivo 1 y 2 son de minimización (Min-Min)*
    - *La función objetivo 1 es de minimización y la 2 es de maximización (Min-Max)*
    - *La función objetivo 1 es de maximización y la 2 es de minimización (Max-Min)*
    - *La función objetivo 1 y 2 son de maximización (Max-Max)*
3. *Adicionar los frentes obtenidos a la lista de frentes.*
4. *Ubicar el depósito como el origen de los ejes coordenados. Sobre los cuatro cuadrantes generados según este origen se aplica los siguientes métodos:*
5. *Para los 4 cuadrantes generados:*
    - *Para el cuadrante I* $\left(\{\forall x_i \mid x_i \geq x_0\} \wedge \{\forall y_i \mid \mathrm{y}_i \geq y_0\}\right)$*, encontrar los frentes suponiendo que es de tipo Max-Max*
    - *Para el cuadrante II* $\left(\{\forall x_i \mid x_i < x_0\} \wedge \{\forall y_i \mid \mathrm{y}_i \geq y_0\}\right)$*, encontrar los frentes suponiendo que es de tipo Min-Max*

- *Para el cuadrante III* $\left(\{\forall x_i \mid x_i < x_0\} \wedge \{\forall y_i \mid y_i < y_0\}\right)$, *encontrar los frentes suponiendo que es de tipo Min-Min*
- *Para el cuadrante IV* $\left(\{\forall x_i \mid x_i \geq x_0\} \wedge \{\forall y_i \mid y_i < y_0\}\right)$, *encontrar los frentes suponiendo que es de tipo Max-Min*

6. *Adicionar los frentes obtenidos a la lista de frentes*
7. *Sí se ha rotado los clientes el número de pasos pre especificado* n_p, *ir al paso 8. De lo contrario rotar cada cliente alrededor del depósito según* n_p *(expresión (2.7) y*
8. *Figura 2.19) y volver al paso 2.*

$$\alpha = \alpha + \frac{2\pi}{n_p} \qquad (2.7)$$

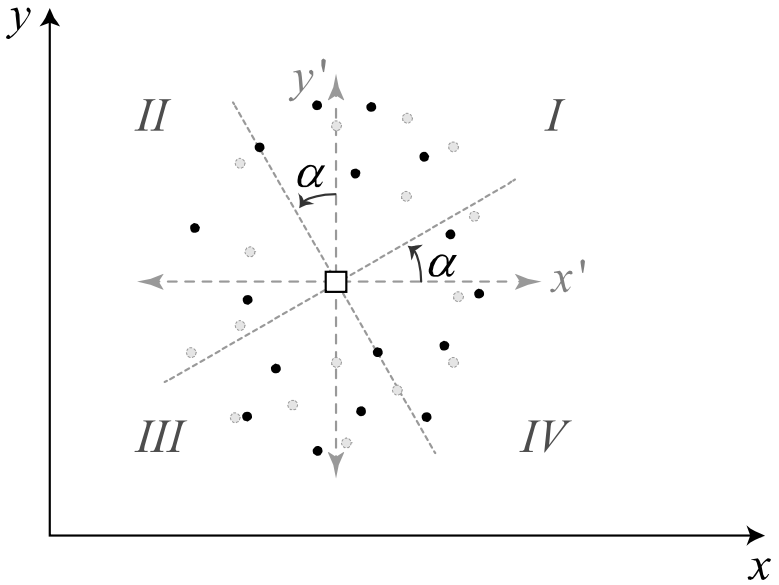


Figura 2.19. Rotación de los clientes alrededor del depósito.

9. *Para toda la lista de frentes obtenidos, calcular Nv*
10. *Calcular Mb, de acuerdo al criterio de aceptación (2.5) y la expresión (2.6)*
11. *Fin de la metodología*

3 Algoritmo de colonia de hormigas (ACO)

3.1 Introducción

Muchos de los algoritmos de cómputo están inspirados en eventos de la naturaleza, es por tal motivo que reciben el nombre de algoritmos bio-inspirados. El algoritmo colonia de hormigas fue propuesto por Dorigo en [16], inspirado en la versión propuesta por Deneuborg en la referencia [27]. Desde entonces son muchas las aplicaciones de dicho algoritmo aplicado a problemas Np-hard, en este caso para el problema de ruteo de vehículos, desde múltiples colonias de hormigas, algoritmos híbridos, diversas maneras de calcular la probabilidad de transición de las hormigas, o la forma como se actualiza la feromona. En este capítulo se explica una metodología del algoritmo de colonia de hormigas basado en la referencia [26], adicionalmente se introducen una serie de mecanismos y procedimientos relacionados con la estrategia de filtro de arcos de la sección anterior, entre otros, los cuales implican algunos cambios respecto al algoritmo tradicional.

3.2 ACO

En base a cuatro experimentos se demostró que las hormigas se comunican de forma indirecta, esto sucedió cuando las hormigas tenían 2 caminos para seleccionar con el fin de llegar al alimento. Las hormigas en un principio iban en un camino u otro de forma aleatoria, sin embargo, conforme estas retornaban a su punto de inicio, las próximas hormigas se desplazaban por el trayecto más corto. Esto sucede porque una hormiga recorre en menor tiempo el camino más corto, por lo tanto la siguiente hormiga en su recorrido se basa en el rastro dejado por su compañero. Este rastro o señal se denomina feromona y es la forma en como las hormigas se comunican, por lo tanto ante los 2 caminos expuestos, la probabilidad de feromona depositada en el camino más corto es más alta, ya que las hormigas que retornan depositando la feromona tardarían menos tiempo en llegar a su colonia, dándoles así una señal de estímulo a las siguientes hormigas que parten del punto de origen.

Esto no quiere decir que las hormigas continúan siempre por el camino más corto, en uno de los experimentos el camino más corto fue retirado, de tal manera que las hormigas solo tenían como única opción el camino largo, luego de un periodo de tiempo el camino corto fue habilitado de nuevo, sin embargo las hormigas continuaron por el camino largo, debido a la concentración de feromona. A partir de esta percepción en el algoritmo colonia de hormigas surge un mecanismo con el fin de

evitar que las hormigas sigan recorriendo siempre el mismo camino, desde el punto de vista de la optimización esto es análogo a un óptimo local, una forma de salir de óptimos locales es a través de la perturbación de la mejor solución actual, para el caso de la colonia de hormigas se habla del fenómeno de la evaporación de la feromona. Si para el experimento anterior la concentración de feromona se evaporara en cierta cantidad conforme pase el tiempo, las probabilidades de que las hormigas circulen por el otro camino, se vuelven altas.

En el algoritmo propuesto por Dorigo en [16], la cantidad de feromona no está asociada al tiempo de transición de las hormigas (en comparación con el modelo propuesto por Deneuborg en [27]), sino de la calidad de la fuente de alimento. El algoritmo básico de colonia de hormigas aplicado al problema de la ruta más corta, se resume, básicamente en moverse de forma probabilística, desde el nodo de origen hasta llegar al nodo final. Como pasos posteriores se debe aplicar el mecanismo de evaporación y depositar la feromona en vista de calidad de la solución obtenida.

3.3 ACO aplicado al problema de CVRP

En algunos se suele representar un vehículo como una hormiga, en este caso una hormiga representa una solución completa del CVRP tal como se muestra en la Figura 3.1.

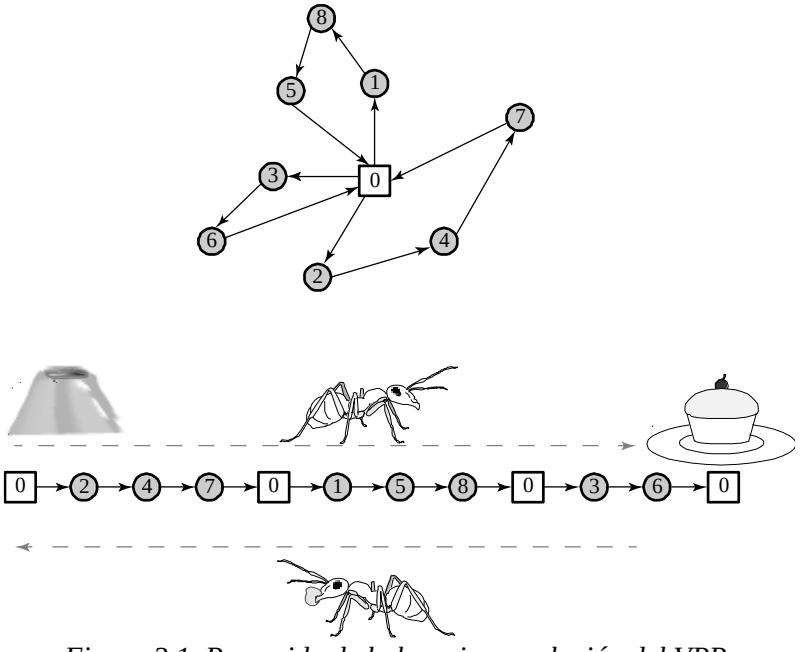


Figura 3.1. Recorrido de la hormiga y solución del VRP.

Dicha solución debe respetar el límite de capacidad de los vehículos. La hormiga decide visitar los vértices en determinado orden de acuerdo a la probabilidad de transición de cada arco, la cual está en función de la feromona (3.1).

$$p_{ij}^{k} = \begin{cases} \dfrac{\tau_{ij}^{\alpha} \cdot \eta_{ij}^{\beta}}{\sum_{k \in N_i^k} \tau_{ik}^{\alpha} \cdot \eta_{ij}^{\beta}} & si \quad j \in N_i^k \\ 0 & si \quad j \notin N_i^k \end{cases} \tag{3.1}$$

Donde N_i^k son el conjunto de nodos vecinos de la hormiga k posicionada en el nodo i, τ_{ij} corresponde a la feromona depositada en el arco ij. η_{ij} Es definido como el inverso de la distancia entre i y j (3.2).

$$\eta_{ij} = \frac{1}{d_{ij}} \tag{3.2}$$

α y β son dos parámetros importantes, ya que el valor de estos indica cuál de los 2 factores $\left(\tau_{ij}, \eta_{ij}\right)$ Involucrados en (3.1) tiene mayor peso o relevancia. Valores altos de alfa indican una probabilidad muy determinista, ya que tendrá mayor dependencia a la distancia entre ambos vértices.

Por lo general la feromona debe ser inicializada en un valor mayor a cero, un caso práctico es que este sea de 1, en [26] se recomienda que tome el valor de la expresión (3.3).

$$\tau_0 = \frac{m}{C^{nn}} \tag{3.3}$$

Donde m es el número de hormigas y C^{nn} corresponde para el caso del agente viajero la distancia más corta obtenida de manera determinista, para el caso del CVRP puede ser el costo de la solución obtenida por algún método heurístico, en este caso el descrito en la sub-sección 3.6.1.

Como se ha mencionado con anterioridad, la feromona depositada depende de la calidad de la solución obtenida.

$$\tau_{ij} = \tau_{ij} + \Delta_{\tau}^{k} \tag{3.4}$$

Para este trabajo se ha definido un valor de delta, el cual depende de la calidad de la solución y del número de vehículos, para esto es necesario definir varios términos. El valor Lk representa el costo total de la solución obtenida, mientras que $l_k(i,j)$ es el costo de la ruta *l*, donde el arco *i,j* está asociado a dicha solución. r_k se define como el número de vehículos empleados en la solución obtenida, r se define como el número de vehículos a emplear para obtener la solución de determinado problema, este valor puede ser mayor o igual al mínimo número de vehículos que se pueden emplear para obtener la solución, este último es un término muy empleado en los modelos del VRP de la referencia [1], esto es:

$$r_{\min} = \left\lceil \frac{D_T}{C} \right\rceil \qquad (3.5)$$

C corresponde a la capacidad de los vehículos del problema y D_T corresponde a la demanda total atendida en el problema de ruteo (3.6).

$$D_T = \sum_{i \in V \setminus 0} d_i \qquad (3.6)$$

Con los anteriores parámetros descritos, el valor Δ_τ^k definido en este trabajo, es dado por la expresión (3.7).

$$\Delta_\tau^k = \left(\frac{l_k(i,j)}{L_k} \right) \left(\frac{r}{r_k} \right) \left(\frac{D_{kl}}{D_T} \right)^a \left(\frac{D_{kl}}{C} \right)^b \qquad (3.7)$$

Dónde L_k es el costo de la solución obtenida de la hormiga *k* con r_k vehículos de capacidad *C* y una demanda total en los clientes de D_T . $l_k(i,j)$ es el costo de la ruta *l* de la hormiga *k*, donde el arco *ij* pertenece a dicha ruta. D_{kl} es la demanda de la ruta *l*, *a* y *b* son números aleatorios entre 0 y 1. En base a la expresión anterior se debe cumplir las propiedades (3.8) -(3.11).

$$r \geq r_{\min} \qquad (3.8)$$

$$l_k(i,j) < L_k \qquad (3.9)$$

$$D_T > C \geq D_{kl} \qquad (3.10)$$

$$\Delta_\tau^k > 0 \qquad (3.11)$$

Con el fin de evitar en caer rápidamente en óptimos locales tal como se menciona en 3.2 ACO, se emplea el mecanismo de evaporación, esta etapa debe realizarse previamente a la actualización de la feromona. La expresión (3.12) incluye el efecto de la evaporación de la feromona.

$$\tau_{ij} = (1-\rho)\tau_{ij} \qquad (3.12)$$

Donde ρ es la tasa de evaporación la cual es dada en el intervalo de cero a uno.

3.4 Algoritmo Básico De Colonia de hormigas

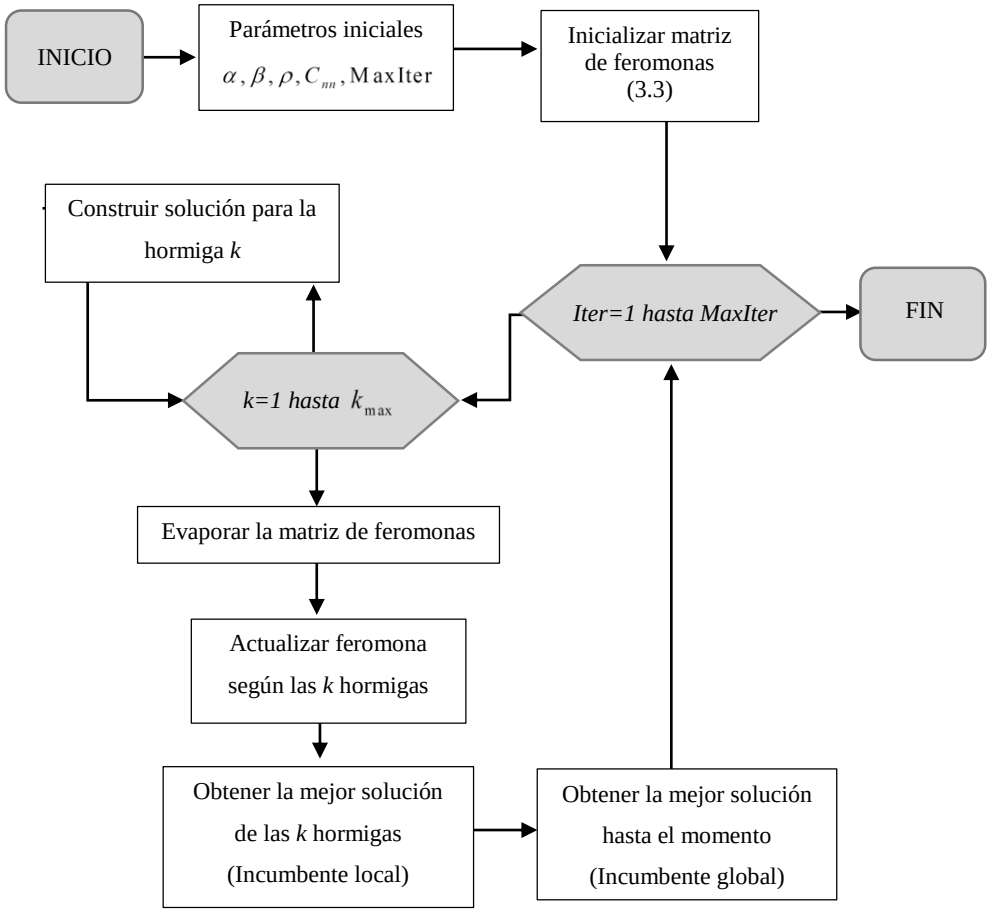


Figura 3.2. Diagrama de flujo del algoritmo colonia de hormigas.

El algoritmo ilustrado en Figura 3.2, corresponde a los pasos básicos para la optimización mediante el algoritmo colonia de hormigas, el procesos de construcción de solución puede contener una mejora a través de una búsqueda local, en las sección posterior se describe las heurísticas empleadas para obtener una solución inicial, la cual indica el valor de C_{nn} .

3.5 Estructura de la solución

Las rutas generadas tiene la siguiente codificación

- Rutas: Es una lista de una estructura denominada ruta
- Demanda Total: Demanda del problema VRP
- Costo total: Costo obtenido en la solución

A su vez el campo rutas se desglosa de la siguiente manera:

- Ruta: Es un vector que representa la ruta realizada por un vehículo
- Demanda: Demanda acumulada en la Ruta
- Costo: Costo de la Ruta

Una solución del problema VRP es de la estructura tipo THormiga, la colonia de hormigas está compuesta por una lista de tipo THormiga:

Tipos:
TRuta: *estructura*
Ruta: *Arreglo de tipo entero;*
Demanda: *Valor de tipo numérico;*
Costo: *Valor de tipo numérico;*
Fin TRuta;

THormiga: *estructura*
Rutas: *Lista de* ***TRuta;***
DemandaTotal: *Valor de tipo numérico;*
CostoTotal: *Valor de tipo numérico;*
Fin THormiga;

Variable:
Colonia*: Lista de tipo* ***THormiga****;*

El ejemplo de la Tabla 3.1 muestra el contenido de una hormiga en particular, en este caso se asume que es la hormiga i de la colonia y cuya solución corresponde a la de la Figura 3.1.

Colonia(i).Rutas(0).Ruta:	2	4	7
Colonia(i).Rutas(1).Ruta:	1	5	8
Colonia(i).Rutas(2).Ruta:	3	6	

Colonia(i).Rutas(0).Demanda:	10
Colonia(i).Rutas(0). Demanda:	8
Colonia(i).Rutas(0). Demanda	9

Colonia(i).Rutas(0).Costo:	100
Colonia(i).Rutas(0).Costo:	150
Colonia(i).Rutas(0).Costo:	80

Tabla 3.1. Contenido de la hormiga i.

3.6 Metodologías para la construcción de soluciones del problema del VRP

Las 2 heurísticas descritas a continuación permiten obtener un valor de la incumbente inicial, las soluciones construidas no tienen en cuenta el mínimo número de vehículos o el número predeterminado de vehículos, sin embargo para la heurística H2, las probabilidades de transición del método ACO son empleadas, estas probabilidades dependen de la feromona (expresión), estás a su vez depende de la cantidad de vehículos empleados en la solución (expresión y), se puede concluir que a menor número de vehículos obtenidos en la solución, mayor es la cantidad de feromona depositada.

3.6.1 Heurística de inserción H1

Esta heurística tiene como función crear una solución de manera rápida, como se menciona con anterioridad, el costo obtenido es empleado como C_{nn}, el procedimiento general es descrito a continuación

1. *Seleccionar como nodo Actual i el depósito, se genera una nueva ruta (ruta Actual).*
2. *De la lista de nodos vecinos al nodo i (N_i^k), seleccionar el nodo más cercano que no haya sido seleccionado previamente j^* (el depósito puede ser seleccionado múltiples veces).*
3. *Si el nodo seleccionado es el depósito, se procede a generar una nueva ruta, empezando de nuevo con el paso 1.*
4. *Si la demanda de la ruta actual se le adiciona la demanda del nodo seleccionado y este excede la capacidad del vehículo, entonces, dicho nodo no es adicionado a la ruta actual (la ruta finaliza su recorrido) y se debe retornar al paso 1, de lo contrario procede con el siguiente paso*
5. *El nodo j^* es ahora el nodo Actual $i = j^*$.*
6. *Si todos los clientes fueron visitados, retornar al paso 2.*

En este caso se considera que todo los nodos son vecinos (grafo completo), para la siguiente heurística se restringe los nodos vecinos, de acuerdo a la metodología de reducción del espacio de búsqueda del capítulo 2.

3.6.2 Heurística de inserción aleatoria H2

Es una heurística similar a H1, los pasos se describen a continuación

1. *Seleccionar de forma aleatoria un vértice que puede incluir el depósito (Nodo Actual) y crear una ruta (Ruta Actual).*
2. *Si el vértice seleccionado es diferente al depósito actualizar la demanda y el costo de la Ruta Actual, la demanda y costo total.*
3. *Seleccionar el Nodo Destino (del conjunto N_i) mediante el método descrito en 3.6.4.*
4. *Si el Nodo Destino es el depósito, se actualiza el costo de la Ruta Actual, el costo total de la solución, se actualiza el Nodo Actual con el valor del Nodo Destino, una nueva ruta es creada (Ruta Actual). Finalmente se retorna al paso 3. En caso contrario ir al siguiente paso.*
5. *Si el Nodo Destino es diferente al depósito, agregar dicho nodo a la Ruta Actual, actualizar la demanda y el costo de la Ruta Actual, la demanda y costo total.*

3.6.3 Nodos vecinos

Hasta el momento se ha mencionado el término vecindario N_i^k, este corresponde a los vértices en los cuales hay opciones de conectividad con el nodo *i*, según la metodología del capítulo 2. Para el problema del CVRP, el grafo es completo, esto quiere decir que cada nodo tiene conectividad con los nodos restantes, sin embargo, para este caso la conectividad de los nodos está restricta por la matriz *Mb* (capítulo 2). Para el nodo *i*, en un principio son nodos vecinos *v*, aquellos elementos de la matriz *Mb* que sean iguales a uno, esto es $m_{iv} = 1$. Esta no es la única condición, se debe cumplir adicionalmente que cualquiera de estos nodos *v*, al ser adicionados no violan la restricción de la capacidad del vehículo. El ejemplo de la Figura 3.3 ilustra el vecindario obtenido para el nodo *i*.

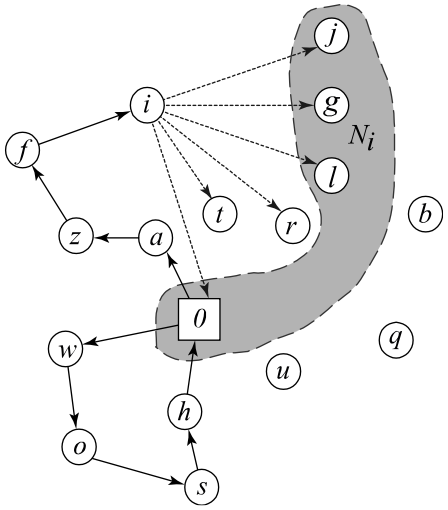


Figura 3.3. Ejemplo de nodos vecinos.

Para este ejemplo se cumple que $m_{ik} = 1 \quad \forall k = 0, g, l, j, r, t$. La demanda acumulada en la ruta está dada por $d = d_a + d_z + d_f + d_i$, la cual cumple $d \leq C$ donde C es la capacidad del vehículo. Se observa que el vecindario de *i* (N_i) está conformado únicamente por los nodos 0, *j, g, l*. Esto sucede porque la demanda de los nodos *r, t* exceden el límite de capacidad del vehículo, esto es: $d + d_r > C \quad \vee \quad d + d_t > C$, mientras que: $d + d_j \leq C \quad \vee \quad d + d_g \leq C \quad \vee \quad d + d_l \leq C$.

3.6.4 Selección del nodo Destino

En esta sub-sección se describe el procedimiento para seleccionar el nodo destino, el cual depende del vecindario del Nodo Actual y de las probabilidades de transición.

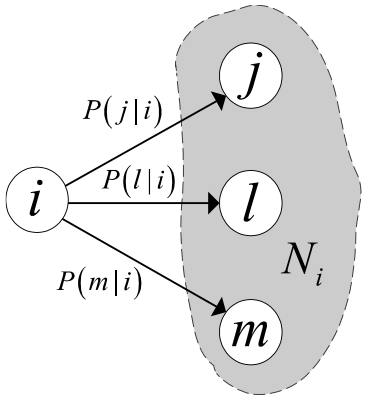


Figura 3.4. Cadena de Markov.

La Figura 3.4 muestra la representación en cadenas de Markov para el vecindario del vértice i, donde $P(j|i)$ es la probabilidad de presentarse el evento j una vez se haya dado el evento i (denominada probabilidad condicionada), de lo anterior se cumple la siguiente propiedad:

$$P(j|i)+P(l|i)+P(m|i)=1 \qquad (3.13)$$

Estas probabilidades condicionadas son las mismas probabilidades obtenidas en la expresión (3.1), esto es:

$$p_{ij}^{k}=P^{k}(j|i) \qquad (3.14)$$

Se debe recordar que el super-índice hace alusión a la hormiga *k*. Las probabilidades de la expresión (3.1), también cumplen con la propiedad expuesta en (3.13). El algoritmo colonia de hormigas expuesto en este capítulo emplea la metodología de selección por ruleta el cual emplea las probabilidades condicionadas y la probabilidad acumulada para determinar el próximo vértice.

3.6.4.1 Método de selección por ruleta

Es un método generalmente usado en algoritmos genéticos. Partiendo de N individuos o vecinos, se realiza los siguientes pasos:

1. *Obtener la lista ordenada de los vecinos con sus probabilidades (en este caso ordenada de forma ascendente de las probabilidades).*
2. *Determinar la probabilidad acumulada:*

$$q_j=\sum_{i=1}^{j}p_i \quad j=1,\cdots,N \qquad (3.15)$$

   Dónde los índices i, j se refieren al orden de cada probabilidad en la lista.

3. *Fijar $j=1$ y fijar x en un valor aleatorio entre 0 y 1, con distribución de probabilidad uniforme.*
4. *Sí se cumple la desigualdad (3.14), el proceso finaliza y el vecino j es seleccionado, de lo contrario ir al paso 5.*

$$X \leq q_j \qquad (3.16)$$

5. *Actualizar,* $j = j + 1$ *y regresar al paso 4.*

Ejemplo:

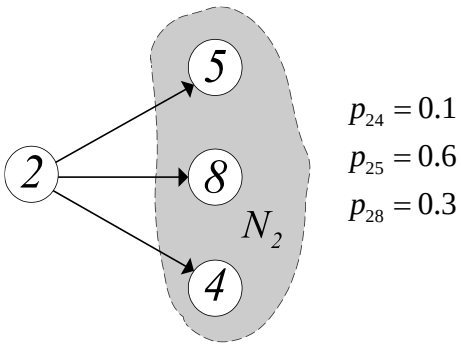


Figura 3.5. Ejemplo de selección por ruleta.

1. Lista de nodos vecinos:

j	*vértice*	p_j
1	*4*	*0.1*
2	*8*	*0.3*
3	*5*	*0.6*

2. Probabilidades acumuladas:

j	*vértice*	p_j	q_j
1	*4*	*0.1*	*0.1*
2	*8*	*0.3*	*0.1+0.3=0.4*
3	*5*	*0.6*	*0.4+0.6=1*

3. Se inicia j y se obtiene el valor de X.

$$j = 1$$
$$X = 0.75$$

4. El proceso cíclico de los pasos 4-5 del algoritmo se repiten hasta llegar a $j = 3$, ya que esta cumple con la restricción (3.16). el algoritmo finaliza entregando el vértice 5 como nodo seleccionado.

3.7 Mejoras locales

Las mejoras locales implementadas consisten en el método de inserción intra-ruta e inter-rutas, 2 opt inter-intra ruta y una mejora local basada en el algoritmo de ahorros de C&W.

Inserción: Este consiste en seleccionar un vértice de una ruta ya conformada (*i*) y cambiarla de posición a lo largo de su propia ruta (intra-ruta) o ubicarlo en otra ruta (inter-rutas). El ejemplo de la Figura 3.7 ilustra el cambio realizado por la mejora de inserción.

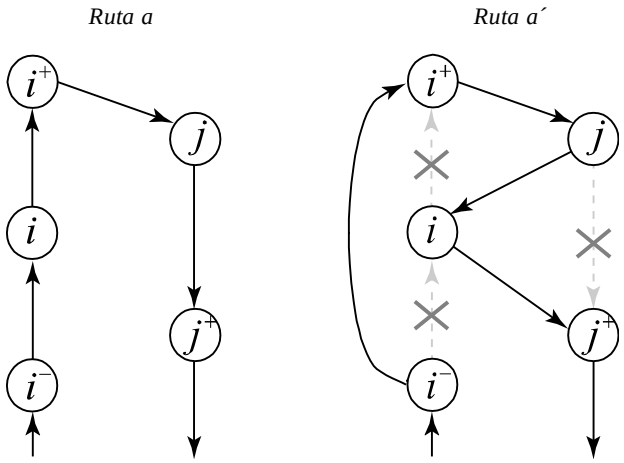


Figura 3.6. Mejora inserción intra-ruta. (a) antes de la mejora, (b) después de la mejora

.

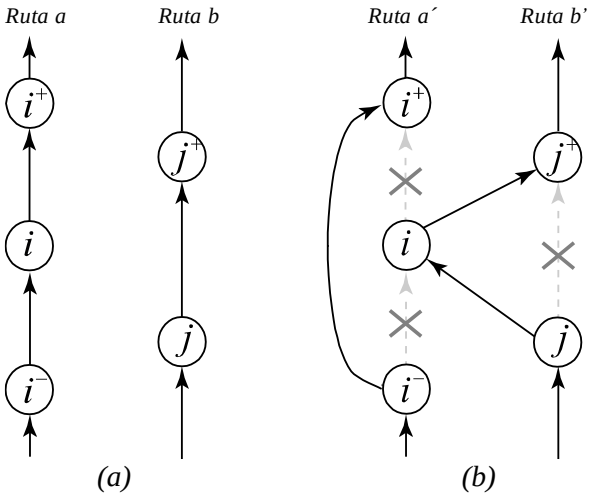


Figura 3.7. Mejora inserción inter-ruta. (a) antes de la mejora, (b) después de la mejora.

El costo de la nueva solución ($C_{T'}$) está dado por (3.17). La disminución del costo (Δ_T) está dado por la diferencia del costo de la ruta antes (C_T) y después ($C_{T'}$) de la mejora (3.18). La mejora inserción se aplica si hay una disminución Δ_T del costo total de la solución, esto sucede sí $\Delta_T > 0$.

$$C_{T'} = C_T - c_{i^- i} - c_{ii^+} - c_{jj^+} + c_{ji} + c_{ij^+} + c_{i^- i^+} \quad (3.17)$$

$$\Delta_T = C_T - C_{T'} = c_{i^- i} + c_{ii^+} + c_{jj^+} - c_{ji} - c_{ij^+} - c_{i^- i^+} \quad (3.18)$$

2-OPT: esta mejora puede ser intra-ruta e inter-ruta, consiste en eliminar de 2 arcos para adicionar 2 nuevos, tal como se ilustra en la Figura 3.8 y Figura 3.9.

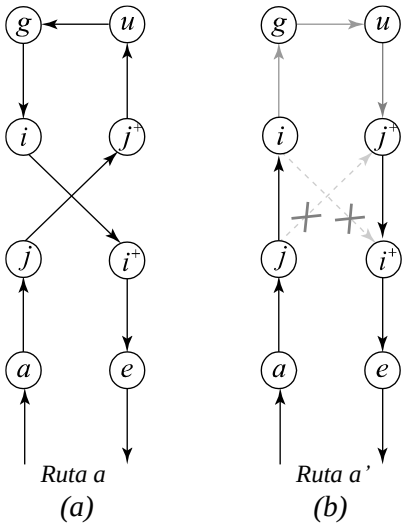


Figura 3.8. Mejora 2-OPT intra-ruta. (a) antes de la mejora, (b) después de la mejora.

De manera similar a la mejora anterior, la mejora 2-OPT se aplica si hay una disminución en el costo total de la solución $(\Delta_T > 0)$. El costo de la nueva ruta se muestra en (3.17), en base a esta se obtiene la disminución del costo total de la solución (3.20).

$$C_{T'} = C_T + c_{ji} + c_{j^+ i^+} - c_{ii^+} - c_{jj^+} \quad (3.19)$$

$$\Delta_T = C_T - C_{T'} = c_{ii^+} + c_{jj^+} - c_{ji} - c_{j^+ i^+} \quad (3.20)$$

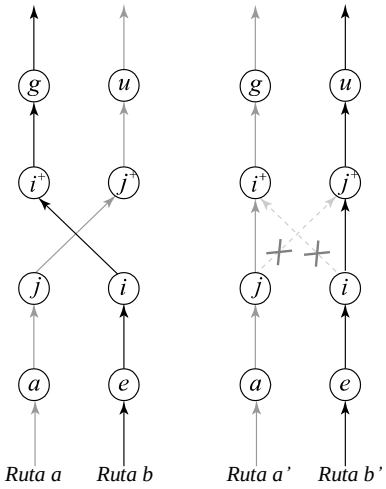
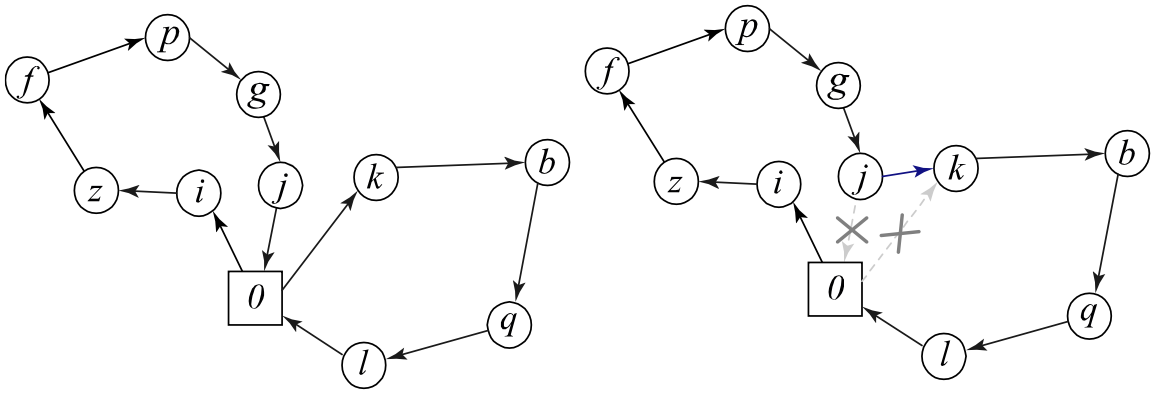


Figura 3.9. Mejora 2-OPT inter-ruta. (a) antes de la mejora, (b) después de la mejora.

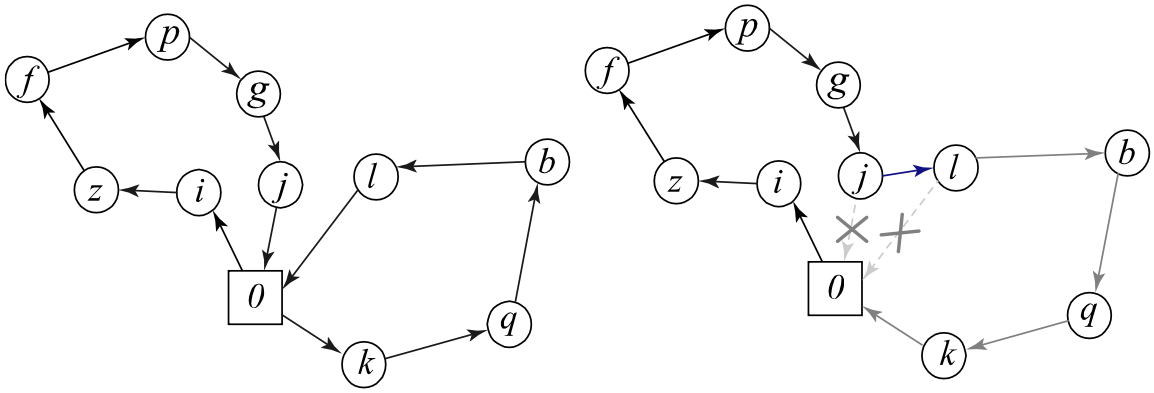
Ahorros: Esta mejora es únicamente inter-ruta, está basado en el algoritmo de ahorros C&W. El proceso consiste en unir 2 rutas por cualquiera de sus puntos extremos, siempre y cuando se presente una disminución del costo total de la solución y no se exceda la capacidad del vehículo. Suponiendo que *i* es el primer nodo de la primera ruta y *j* el último nodo, *k* es el primer nodo de la segunda ruta y *l* el último nodo. Como hay posibilidad de unir las rutas por cualquiera de estos puntos extremos, se tienen entonces 4 casos, los cuales son ilustrados en Figura 3.10-Figura 3.13.

En este caso el problema es simétrico, el costo c_{ij} es igual al costo c_{ji}, por lo tanto para los casos de la Figura 3.11 y Figura 3.12, en los que implica invertir la ruta no surge una variación en el costo debido a esta inversión. Partiendo de *a* (punto extremo de la primera ruta) y *b* (punto extremo de la segunda ruta) como los puntos extremos a través de los cuales se unen las rutas Figura 3.14, los cuales pueden tomar los valores de *i,j* o *k,l* según los casos ilustrativos, se define el costo de la nueva ruta según la expresión (3.21). El ahorro S_{ab} (3.22) se obtiene a partir de dicha expresión y de manera similar a las mejoras anteriores, el ahorro debe ser mayor a cero si existe una mejora en el costo total de la solución.



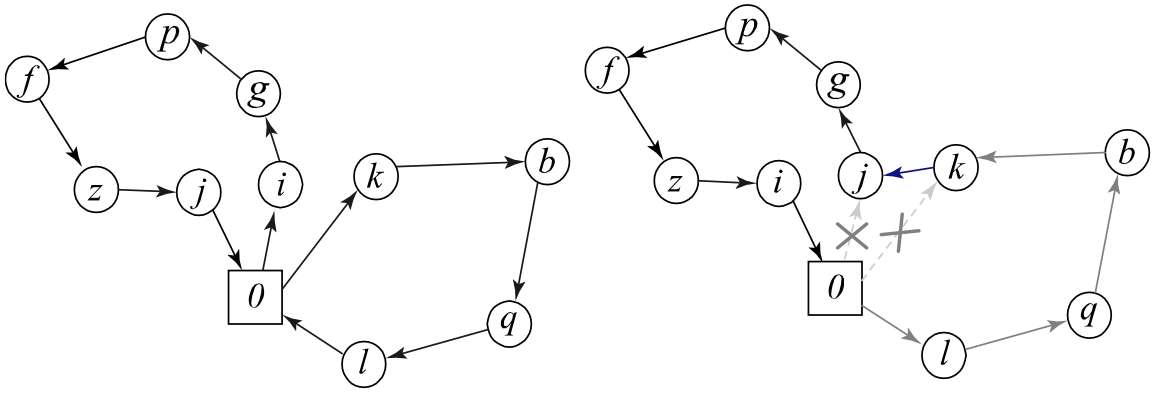
(a) *(b)*

Figura 3.10. Mejora C&W caso 1. (a) antes de la mejora, (b) después de la mejora.



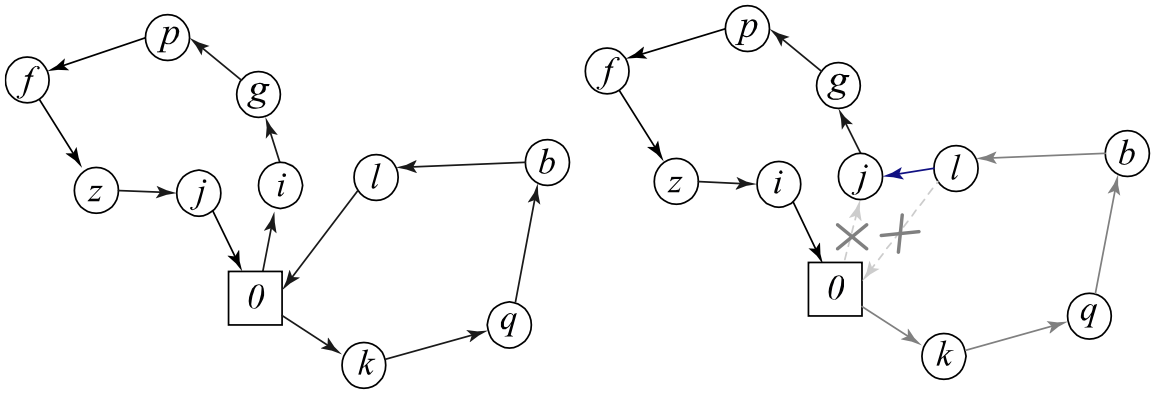
(a) *(b)*

Figura 3.11. Mejora C&W caso 2. (a) antes de la mejora, (b) después de la mejora.



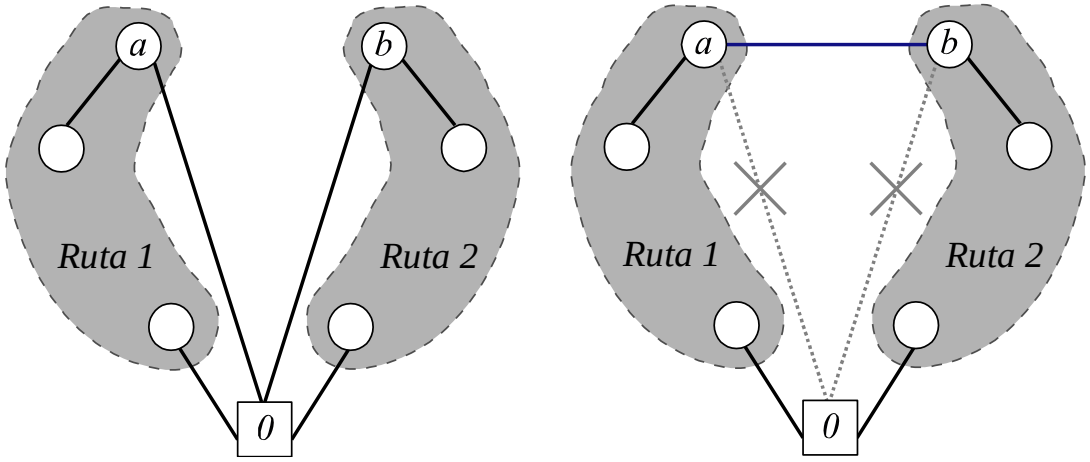
(a) *(b)*

Figura 3.12. Mejora C&W caso 3. (a) antes de la mejora, (b) después de la mejora.



(a) *(b)*

Figura 3.13. Mejora C&W caso 4. (a) antes de la mejora, (b) después de la mejora.



(a) *(b)*

Figura 3.14. Mejora C&W forma generalizada. (a) antes de la mejora, (b) después de la mejora.

$$C_{T'} = C_T + c_{ab} - c_{a0} - c_{b0} \quad (3.21)$$

$$S_{ab} = C_T - C_{T'} = c_{a0} + c_{b0} - c_{ab} \quad (3.22)$$

Hay 3 modos de implementar las mejoras:

- Primera mejora: El método finaliza luego de encontrar una mejora.
- Mejor mejora: El método prueba de forma exhaustiva los cambios posibles, el algoritmo finaliza y elige únicamente aquel cambio que produce mayor mejora en la función objetivo

(para problemas de minimización, aquel que presenta mayor disminución en la función objetivo).

- Mejoras secuenciales: se recorre todas las posiciones actuales de las rutas evaluando las posibles mejoras. Cada vez que se presente la mejora en el costo, se implementa el cambio en las rutas asociadas.

3.7.1 Etapas de mejoría local

En este trabajo se incorporan las mejoras locales en 2 etapas:

1. La primera es en la construcción de la solución (hormiga), todas las mejoras de esta etapa se hacen sobre el vecindario obtenido de acuerdo a la reducción del espacio de búsqueda (Frentes no dominados). En esta etapa se incorpora las siguientes mejoras en el orden especificado:
    - Mejora Intra ruta 2-OPT en modo Secuencial
    - 2 vecese la mejora Inserción Inter-Itra ruta en modo Secuencial
    - Mejora de Ahorros (Inter ruta por defecto) en modo Secuencial
2. En la segunda etapa se realiza las mejoras sobre la Incumbente Global cada 100 iteraciones, en el orden especificado:
    - La mejora Intra ruta 2-OPT en modo Secuencial, teniendo en cuanta los arcos activos, de acuerdo a la reducción del espacio de búsqueda
    - La mejora Inter-Intra ruta 2-OPT en modo Secuencial, teniendo en cuenta todos los arcos de la instancia
    - Mejora Inserción Inter-Itra ruta en modo Secuencial, teniendo en cuenta todos los arcos de la instancia
    - Mejora de Ahorros (Inter ruta por defecto) en modo “Primera Mejora”, teniendo en cuanta los arcos activos, de acuerdo a la reducción del espacio de búsqueda

3.8 Aco Modificado

Finalmente luego de todos los procedimientos descritos, se implementa una modificación al ACO planteado en las secciones 3.3-3.4, es preciso destacar que la actualización de la feromona y el vecindario de cada vértice, son un cambio al método tradicional. La evaluación de la feromona busca

apremiar aquellas soluciones que resultan con un número de vehículos menor o igual número de vehículos prestablecido ($r_k \leq r$).

Este algoritmo busca soluciones en la infactibilidad, respecto a la cantidad de vehículos. A medida que las iteraciones avanzan, la solución es menos infactible. Se dice que una solución es mejor que otra si esta tiene menor cantidad de vehículos. En caso de tener la misma cantidad, se dice que es mejor si tiene menor costo.

Con la metodología planteada se pretende encontrar soluciones sobre los arcos filtrados, evitando emplear todos los arcos del sistema. Esto es análogo al algoritmo de búsqueda tabú que penaliza o sanciona ciertos arcos con el fin de evitar caer en óptimos locales. El algoritmo de colonia de hormigas se considera un método de exploración en los problemas de optimización, como método de explotación se implementa la búsqueda local, esta se hace sobre la colonia, cuando se construye cada solución. Adicionalmente se realiza una serie de procedimientos cada determinado ciclo, estos son:

1. Búsqueda local sobre la incumbente global sin tener en cuenta la restricción de los arcos impuesta por el método de filtro de arcos expuesto en el capítulo 2. Este paso es importante ya que la metodología expuesta en este capítulo, revela que para las instancias probadas, no todos los arcos de la solución óptima global son encontrados mediante este método. Por lo tanto dicha búsqueda local puede encontrar arcos que pertenecen a la solución óptima.
2. Algunas referencias proponen reiniciar la matriz de feromona para explorar nuevas soluciones, en este trabajo, en vez de reiniciar dicha matriz, se realiza una evaporación adicional con una tasa de evaporación muy elevada (evaporación forzada). Esta se realiza cuando la incumbente global no mejora en un 5% durante 100 iteraciones sucesivas.
3. Almacenar lista élite. Este paso es importante para la siguiente etapa descrita en el capítulo 7, el cual consiste en aplicar el modelo de “Set Partition” sobre esta lista. La lista almacena la incumbente local y la solución de menor costo (puede ser factible o infactible) durante las primeras 250 iteraciones, después las almacena cada 5 iteraciones. Estas se almacenan si tienen una diversidad en el costo del 5% respecto a la última solución almacenada en dicha lista. Adicionalmente se almacena la incumbente global si esta mejora con la búsqueda local (especificada en el paso 1) que tiene en cuanta todos los arcos.

En la Figura 3.15 y Figura 3.16 se describe el algoritmo propuesto que combina las metodologías descritas en 2-3.

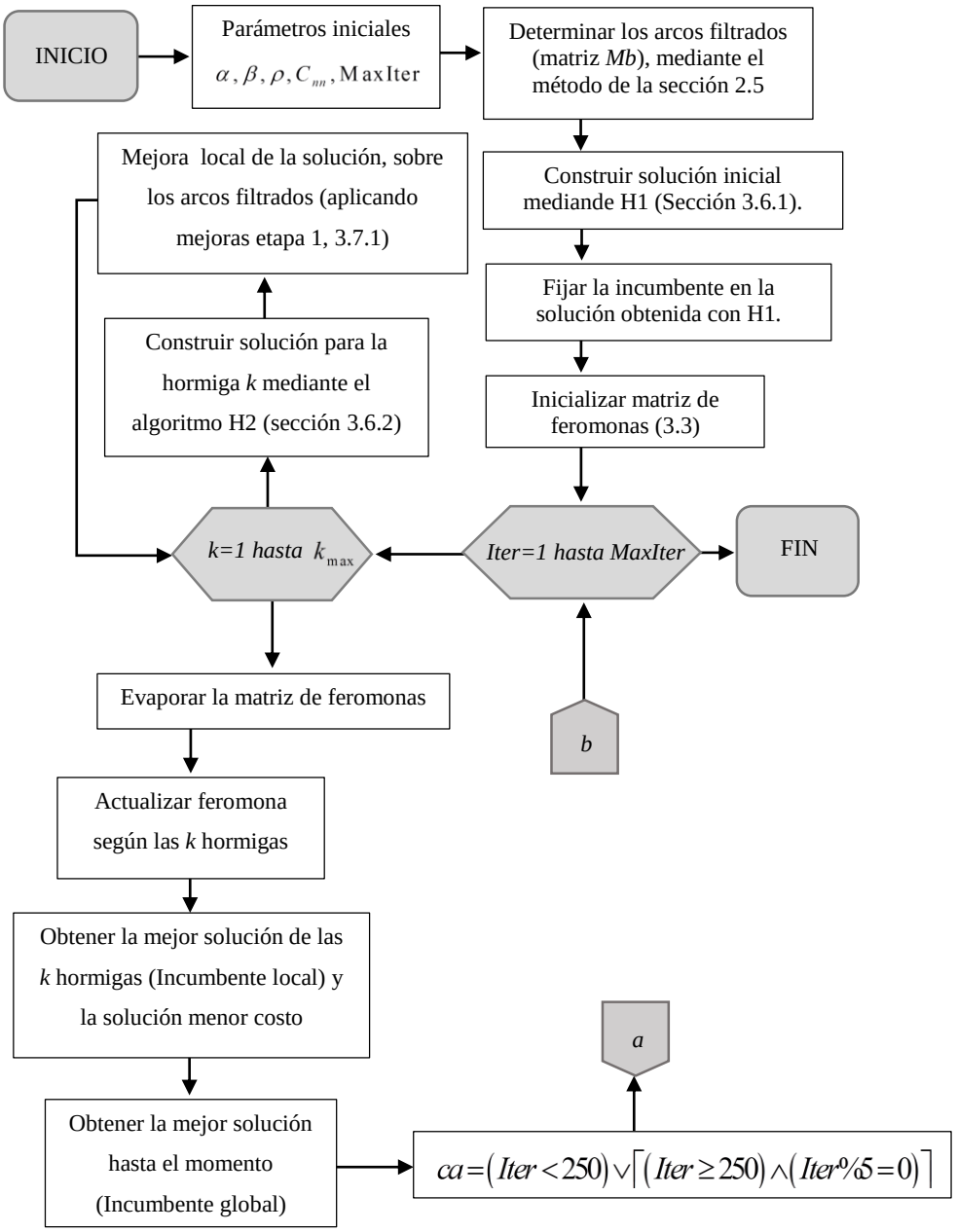


Figura 3.15. Diagrama de flujo algoritmo FA+ACO Modificado parte 1.

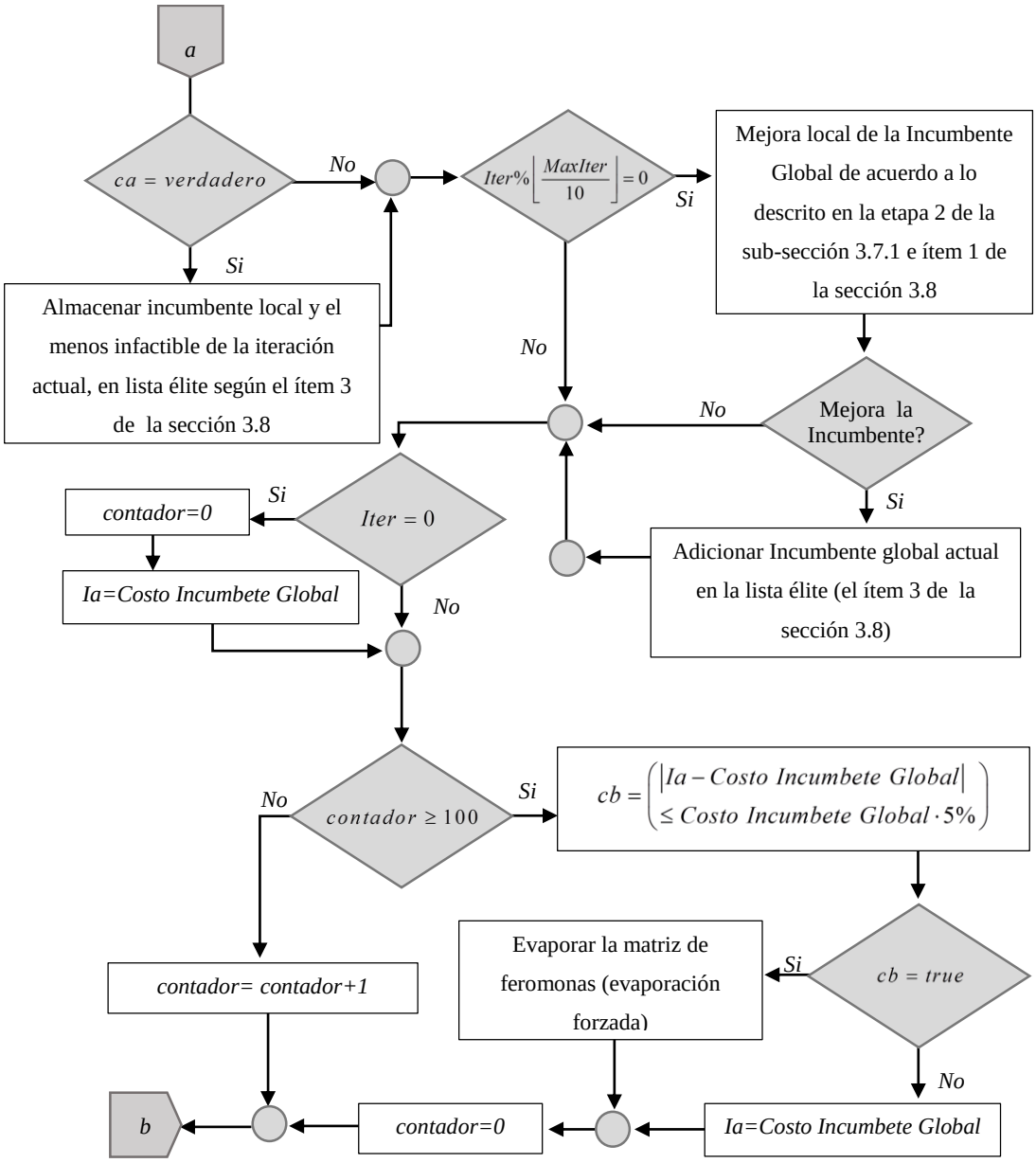


Figura 3.16. Diagrama de flujo algoritmo FA+ACO Modificado parte 2.

4 Método dual simplex canalizado

4.1 Introducción

La metodología expuesta es consultada en las referencias [28] y [29], para una mayor información es necesario consultar dichas referencias.

El método dual simplex canalizado resuelve problemas de la forma

$$\min\{z = cx\}$$

s.a:

$$Ax = b \qquad (4.1)$$

$$l \le x \le u$$

El método dual simplex canalizado se mueve a través de los puntos extremos (PE) no factibles del problema hasta llegar a una solución básica factible (SBF) la cual resulta ser el óptimo del problema. Se debe tener en cuenta las siguientes consideraciones:

1. Las variables básicas asumen un valor entre su límite inferior y superior

$$l_B \le x_B \le u_B \qquad (4.2)$$

2. Las variables no básicas se encuentran en su límite inferior o superior

$$x_N = l_B \quad \wedge \quad x_N = u_B \qquad (4.3)$$

   De acuerdo al segundo ítem, se definen 2 subconjuntos:

   R_1: Conjunto de variables no básicas en su límite inferior

   R_2: Conjunto de variables no básicas en su límite superior

3. El método dual canalizado debe partir de una base óptima. Para el cuadro dual simplex canalizado, se satisface esta condición si se cumple los criterios establecidos en (4.4).

$$\{y_{0j} \le 0 \quad | \, j \in R_1\} \quad \wedge \quad \{y_{0j} \ge 0 \quad | \, j \in R_2\} \qquad (4.4)$$

El cuadro simplex implementado corresponde al cuadro simplex Garfinkel-Nemhauser. El cual parte de expresar las variables básicas en términos de las no básicas. En este caso se parte de la siguiente descomposición:

$$A=\begin{bmatrix} B & N_1 & N_2 \end{bmatrix} \quad x=\begin{bmatrix} x_B \\ x_{N_1} \\ x_{N_2} \end{bmatrix} \quad C=\begin{bmatrix} C_B & C_{N_1} & C_{N_2} \end{bmatrix} \tag{4.5}$$

La función objetivo está dado por:

$$z=\begin{bmatrix} C_B & C_{N_1} & C_{N_2} \end{bmatrix}\begin{bmatrix} x_B \\ x_{N_1} \\ x_{N_2} \end{bmatrix} \tag{4.6}$$

Al expresar las restricciones y la función objetivo, en función de x_{N_1} y x_{N_2} se tiene:

$$\begin{bmatrix} z \\ x_B \end{bmatrix}=\begin{bmatrix} c_B B^{-1}b \\ B^{-1}b \end{bmatrix}-\begin{bmatrix} c_B B^{-1}N_1 - c_{N_1} \\ B^{-1}N_1 \end{bmatrix}x_{N_1}-\begin{bmatrix} c_B B^{-1}N_2 - c_{N_2} \\ B^{-1}N_2 \end{bmatrix}x_{N_2} \tag{4.7}$$

Como las variables no básicas se encuentran en su límite inferior o superior; las variables básicas y la función objetivo pueden ser determinadas según la expresión (4.7), tal como se muestra en (4.8).

$$\begin{bmatrix} z \\ x_B \end{bmatrix}=\begin{bmatrix} c_B B^{-1}b \\ B^{-1}b \end{bmatrix}-\begin{bmatrix} c_B B^{-1}N_1 - c_{N_1} \\ B^{-1}N_1 \end{bmatrix}l_{N_1}-\begin{bmatrix} c_B B^{-1}N_2 - c_{N_2} \\ B^{-1}N_2 \end{bmatrix}u_{N_2} \tag{4.8}$$

El cuadro simplex se construye en base a las expresiones (4.7) y (4.8).

	RHS	R_1 $-x_{N_1}$	R_2 $-x_{N_2}$
z	$c_B B^{-1}b-\left[c_B B^{-1}N_1-c_{N_1}\right]\left[l_{N_1}\right]-\left[c_B B^{-1}N_2-c_{N_2}\right]\left[u_{N_2}\right]$	$c_B B^{-1}N_1-c_{N_1}$	$c_B B^{-1}N_2-c_{N_2}$
x_B	$B^{-1}b-\left[B^{-1}N_1\right]\left[l_{N_1}\right]-\left[B^{-1}N_2\right]\left[u_{N_1}\right]$	$B^{-1}N_1$	$B^{-1}N_2$

Tabla 4.1 Cuadro Dual simplex Canalizado.

Haciendo un cambio de notación, este puede ser expresado tal como se muestra en Tabla 4.2.

		RHS	$-x_j$	…	$-x_k$	…	$-x_l$
	z	z_{00}	y_{0j}	…	y_{0k}	…	y_{0l}
m	x_{Bi}	z_{i0}	y_{ij}	…	y_{ik}	…	y_{il}
	⋮	⋮	⋮	…	⋮	…	⋮
	x_{Br}	z_{r0}	y_{rj}	…	y_{rk}	…	y_{rl}
	⋮	⋮	⋮	…	⋮	…	⋮
	x_{Bm}	z_{m0}	y_{mj}	…	y_{mk}	…	y_{ml}

Tabla 4.2 Cuadro Dual simplex Canalizado (Cambio de notación).

Dónde x_{Bi} (4.9) se obtiene de la expresión (4.7) .

$$x_{Bi} = z_{i0} = y_{i0} - \sum_{j \in R_1} y_{ij} \cdot x_j - \sum_{j \in R_2} y_{ij} \cdot x_j$$
$$x_j = l_1 \quad \forall j \in R_1$$
$$x_j = u_2 \quad \forall j \in R_2 \qquad (4.9)$$

4.2 Actualizar los elementos del cuadro Dual simplex canalizado

Para actualizar los elementos del cuadro dual simplex canalizado se parte de la fila pivote *i* y la columna pivote *j*. Los elementos diferentes a la columna *RHS* se actualizan de igual forma que en los métodos dual simplex y primal simplex, esto se muestra en las expresiones (4.10)-(4.12).

$$y_{ij} = y_{ij} - \frac{y_{ik} \cdot y_{rj}}{y_{rk}} \quad \left\{ \begin{array}{lll} \forall i = 0, \cdots, m & \wedge & i \neq r \\ \forall j = 1, \cdots, n & \wedge & j \neq k \end{array} \right\} \qquad (4.10)$$

Para la columna pivote se obtiene:

$$y_{ik} = \frac{-1}{y_{ik}} \quad \forall i = 0, \cdots, m \quad \wedge \quad i \neq r \qquad (4.11)$$

La fila pivote, incluyendo el elemento pivote (y_{rk}) se actualiza según (4.12).

$$y_{rj} = \frac{1}{y_{rj}} \quad \forall j = 0,\cdots,n \tag{4.12}$$

La Tabla 4.3 muestra una forma fácil de saber cuáles elementos son participes en la actualización de los y_{ij}. Esto es únicamente para los elementos diferentes de y_{rj} y y_{ik}.

	RHS	$-x_j$	…	$-x_k$	…	$-x_l$
z	z_{00}	y_{0j}	…	y_{0k}	…	y_{0l}
x_{Bi}	z_{i0}	y_{ij}	…	y_{ik}	…	y_{il}
	⋮	⋮	…	⋮	…	⋮
x_{Br}	z_{r0}	y_{rj}	…	y_{rk}	…	y_{rl}
	⋮	⋮	…	⋮	…	⋮
x_{Bm}	z_{m0}	y_{mj}	…	y_{mk}	…	y_{ml}

Tabla 4.3 Actualizar Cuadro Dual simplex Canalizado (Cambio de notación).

4.3 Variable básica candidata a salir de la base

La variable básica candidata a salir de la base es aquella que este por fuera de sus límites, por lo tanto hay dos posibles casos como ya se ha mencionado, para aquellas que están por encima de su límite superior y para aquellas que están por debajo de su límite inferior. La variable que sale de la base es aquella que más viole su límite o cota (inferior o superior) (4.13).

$$x_{Br} \rightarrow \max\left\{\left(z_{i0} - u_i\right) \quad \forall i / z_{i0} > u_i \quad ; \quad \left(l_i - z_{i0}\right) \quad \forall i / z_{i0} < l_i\right\} \tag{4.13}$$

De manera similar, la variable candidata a entrar a la base presenta dos estados, que esté en su límite inferior o que esté en su límite superior. Por lo tanto hay cuatro posibles combinaciones (entre las variables básicas y no básicas), pero en el proceso de cambio de base se presenta uno de estos cuatro posibles casos.

Suponiendo que *r* representa el índice de la variable básica a salir de la base y *k* representa el índice de la variable no básica que va formar parte de la base. Se define un valor $\Delta_k \geq 0$ necesario para establecer el movimiento de las variables básicas de tal manera que no salgan de sus límites o cotas. A partir de estas suposiciones se exponen a continuación los 4 posibles casos en el cambio de baso del cuadro Dual Simplex Canalizado.

- **Caso 1:** La Variable básica viola el límite superior $z_{r0} > u_r$ y la no básica está en su límite superior $x_k = u_k$

Para este caso, la variable no básica que estaba en su límite superior debe disminuir su valor para entrar a la base.

$$\begin{aligned} x_k &= u_k \\ x_k &= x_k - \Delta_k \end{aligned} \qquad (4.14)$$

Dicha disminución produce un cambio en las variables básicas, esto se ilustra al retomar la expresión (4.14) y reemplazar esta en (4.9), tal como se muestra en (4.15).

$$\begin{aligned} x_{Bi} &= z_{i0} = y_{i0} - \sum_{j \in R_1} y_{ij} \cdot x_j - \sum_{\substack{j \in R_2 \\ j \neq k}} y_{ij} \cdot x_j - y_{ik} \cdot x_k \\ x_{Bi} &= y_{i0} - \sum_{j \in R_1} y_{ij} \cdot x_j - \sum_{\substack{j \in R_2 \\ j \neq k}} y_{ij} \cdot x_j - y_{ik} \cdot \left(x_k - \Delta_k \right) \\ x_{Bi} &= y_{i0} - \sum_{j \in R_1} y_{ij} \cdot x_j - \sum_{j \in R_2} y_{ij} \cdot x_j + y_{ik} \cdot \Delta_k \end{aligned} \qquad (4.15)$$

Tomando (4.15), teniendo en cuenta el valor de z_{io}, se obtiene la simplificación mostrada en la expresión (4.16). A partir de esta expresión y teniendo en cuanta a z_{r0} o x_{Br}, se obtiene el valor de Δ_k una vez se lleve al límite superior (u_r) la variable básica x_{Br} al salir de la base (4.17).

$$x_{Bi} = z_{i0} + y_{ik} \cdot \Delta_k \qquad (4.16)$$

$$\begin{aligned} x_{Br} &= z_{r0} + y_{rk} \cdot \Delta_k = u_r \\ \Delta_k &= -\left(\frac{z_{r0} - u_r}{y_{rk}} \right) \end{aligned} \qquad (4.17)$$

Retomando (4.14) y reemplazando (4.17) se obtiene el valor de la nueva variable básica (4.18).

$$\begin{aligned} x_k &= x_k - \Delta_k \\ x_k &= u_k - \frac{u_r - z_{r0}}{y_{rk}} \end{aligned} \qquad (4.18)$$

El resto de variables básicas se determina mediante la expresión (4.16) (teniendo en cuenta la expresión (4.17)), tal como se muestra en (4.19).

$$x_{Bi} = z_{i0} + y_{ik} \cdot \left(\frac{u_r - z_{r0}}{y_{rk}} \right) \quad \forall i \neq r \tag{4.19}$$

La función objetivo asume el valor de la expresión (4.20).

$$x_{00} = z_{00} + y_{0k} \cdot \Delta_k \tag{4.20}$$

De acuerdo a la expresión (4.17), puesto que x_{Br} debe disminuir su valor (ya que está por encima de su límite superior) y Δ_k es mayor a cero, se deduce que y_{rk} debe ser menor que cero para lograr dicha disminución.

- **Caso 2:** La Variable básica viola el límite superior $z_{r0} > u_r$ y la no básica está en su límite inferior $x_k = l_k$.

La variable no básica que estaba en su límite inferior debe aumentar su valor para entrar a la base

$$\begin{aligned} x_k &= l_k \\ x_k &= x_k + \Delta_k \end{aligned} \tag{4.21}$$

El cambio de las variables básicas se ve reflejado al reemplazar (4.21) en (4.9), tal como se muestra en (4.22)

$$\begin{aligned} x_{Bi} = z_{i0} &= y_{i0} - \sum_{j \in R_1} y_{ij} \cdot x_j - \sum_{\substack{j \in R_2 \\ j \neq k}} y_{ij} \cdot x_j - y_{ik} \cdot x_k \\ x_{Bi} &= y_{i0} - \sum_{j \in R_1} y_{ij} \cdot x_j - \sum_{\substack{j \in R_2 \\ j \neq k}} y_{ij} \cdot x_j - y_{ik} \cdot \left(x_k + \Delta_k \right) \\ x_{Bi} &= y_{i0} - \sum_{j \in R_1} y_{ij} \cdot x_j - \sum_{j \in R_2} y_{ij} \cdot x_j - y_{ik} \cdot \Delta_k \end{aligned} \tag{4.22}$$

Simplificando se obtiene:

$$x_{Bi} = z_{i0} - y_{ik} \cdot \Delta_k \tag{4.23}$$

Tomando (4.23), el valor de Δ_k se obtiene al llevar la variable básica x_{Br} que abandona la base a su límite superior (4.24).

$$x_{Br} = z_{r0} - y_{rk} \cdot \Delta_k = u_r$$
$$\Delta_k = \frac{z_{r0} - u_r}{y_{rk}} \qquad (4.24)$$

Retomando (4.21) y reemplazando (4.24), se obtiene el valor de la variable no básica que entra a la base (4.25).

$$x_k = x_k + \Delta_k$$
$$x_k = l_k + \frac{z_{r0} - u_r}{y_{rk}} \qquad (4.25)$$

El resto de variables básicas se determina mediante la expresión (4.23) y (4.24), tal como se muestra en (4.26).

$$x_{Bi} = z_{i0} + y_{ik} \cdot \left(\frac{u_r - z_{r0}}{y_{rk}} \right) \quad \forall i \neq r \qquad (4.26)$$

La función objetivo asume el valor de la expresión (4.28).

$$x_{00} = z_{00} - y_{0k} \cdot \Delta_k \qquad (4.27)$$

De acuerdo a la expresión (4.24) y de manera similar al caso 1, debido a que x_{Br} debe disminuir su valor (ya que está por encima de su límite superior) y Δ_k es mayor a cero, entonces el valor de y_{rk} debe ser estrictamente mayor que cero.

- **Caso 3:** La Variable básica viola el límite inferior $z_{r0} < l_r$ y la no básica está en su límite superior $x_k = u_k$

En este caso la variable no básica que está en su límite superior, debe disminuir su valor (4.14), por otro lado la variable básica debe aumentar su valor hasta llegar a su límite inferior.

Retomando las expresiones (4.14), (4.15) y (4.16) se puede obtener el valor de Δ_k necesario para llevar la variable básica (x_{Br}) que saldrá de la base a su límite inferior (4.28).

$$x_{Br} = z_{r0} + y_{rk} \cdot \Delta_k = l_r$$
$$\Delta_k = -\left(\frac{z_{r0} - l_r}{y_{rk}} \right) \qquad (4.28)$$

El valor de la variable no básica que entra a la base x_k , puede ser obtenido a partir de (4.14) , el resto de variables básicas (sin incluir x_{Br}) se determina mediante la expresión (4.16) y la función objetivo mediante (4.20). Adicionalmente se observa que y_{rk} debe ser estrictamente mayor a cero, ya que Δ_k es mayor a cero y x_{Br} debe aumentar su valor (ya que está por debajo de su límite inferior).

- **Caso 4:** La Variable básica viola el límite inferior $z_{r0} < l_r$ y la no básica está en su límite inferior $x_k = l_k$.

Este es el último de los casos, la variable no básica que entra a la base debe aumentar su valor, esto debe producir un aumento en la variable básica que saldrá de la base, tal aumento es hasta su límite inferior.

Para obtener los valores de interés, se parte de (4.21),(4.22) y (4.23). De esta manera se obtiene el valor de Δ_k necesario para llevar la variable básica (x_{Br}) (que sale de la base) a su límite inferior.

$$x_{Br} = z_{r0} - y_{rk} \cdot \Delta_k = l_r$$
$$\Delta_k = -\left(\frac{l_r - z_{r0}}{y_{rk}} \right) \qquad (4.29)$$

Respecto a la variable básica que entra a la base, esta puede ser determinada mediante la expresión (4.21), la función objetivo mediante (4.27), el resto de variables básicas son calculadas mediante la expresión (4.23), adicionalmente, para este caso se tiene que y_{rk} debe ser estrictamente menor a cero, ya que Δ_k es mayor a cero y x_{Br} debe aumentar su valor (debido a que está por debajo de su límite inferior).

Finalmente la Tabla 4.4 muestra un resumen de las principales características de los cuatro casos. La tabla muestra cómo se comporta cada una de las variables de interés (x_k , x_{Br} , x_{Bi}), según el caso y los valores de y_{ik} .

Caso \ x		x_k	x_{Br}	x_{Bi} $\forall i \neq r$	
				Sí $y_{ik} < 0$	Sí $y_{ik} > 0$
Caso1: $z_{r0} > u_r \quad \wedge \quad x_k = u_k$	$y_{rk} < 0$	↓	↓	↓	↑
Caso2: $z_{r0} > u_r \quad \wedge \quad x_k = l_k$	$y_{rk} > 0$	↑	↓	↑	↓
Caso3: $z_{r0} < l_r \quad \wedge \quad x_k = u_k$	$y_{rk} > 0$	↓	↑	↓	↑
Caso4: $z_{r0} < l_r \quad \wedge \quad x_k = l_k$	$y_{rk} < 0$	↑	↑	↑	↓

Tabla 4.4. Características de las variables asociadas en el cambio de base.

4.4 Selección de la variable no básica candidata a entrar a la base

La selección de la columna pivote está orientada según las expresiones (4.30) al (4.33), estas garantiza que el cuadro dual simplex canalizado permanezca óptimo.

Caso 1:

$$-\frac{y_{0k}}{y_{rk}} = \text{mín}\left\{-\frac{y_{0j}}{y_{rj}} \quad \forall j \in R_2 \wedge y_{rj} < 0\right\} \tag{4.30}$$

Caso 2:

$$-\frac{y_{0k}}{y_{rk}} = \text{mín}\left\{-\frac{y_{0j}}{y_{rj}} \quad \forall j \in R_1 \wedge y_{rj} > 0\right\} \tag{4.31}$$

Caso 3:

$$\frac{y_{0k}}{y_{rk}} = \text{mín}\left\{\frac{y_{0j}}{y_{rj}} \quad \forall j \in R_2 \wedge y_{rj} > 0\right\} \tag{4.32}$$

Caso 4:

$$\frac{y_{0k}}{y_{rk}} = \text{mín}\left\{\frac{y_{0j}}{y_{rj}} \quad \forall j \in R_1 \wedge y_{rj} < 0\right\} \tag{4.33}$$

4.5 Resumen del Método Dual Simplex Canalizado

En la Figura 4.1 se presenta un esquema organizado de cómo realizar la actualización del cuadro simplex según el caso que se presente.

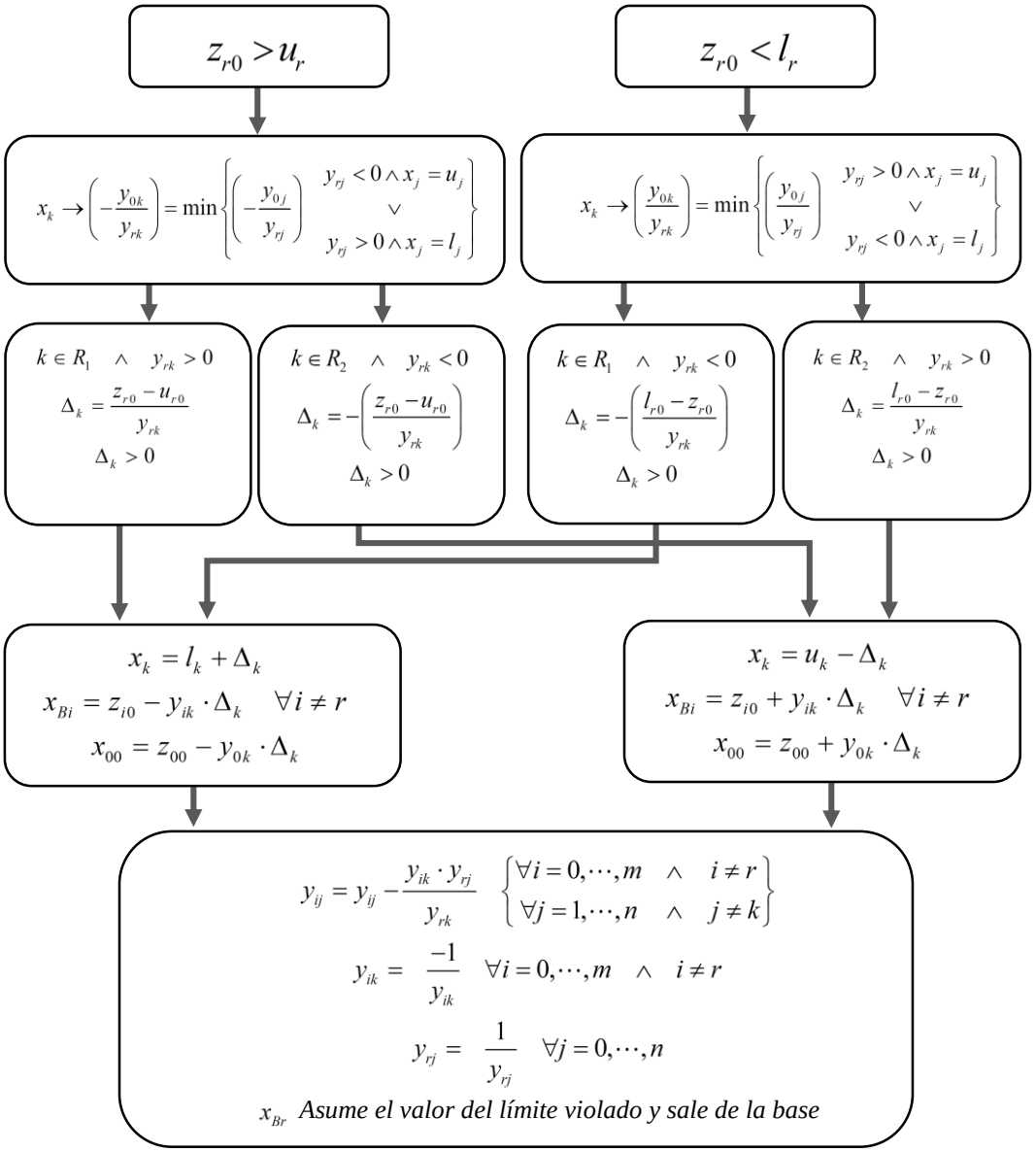


Figura 4.1. Resumen Método Dual Simplex Canalizado.

4.6 Base empleada para el Dual Simplex Canalizado

Se debe seleccionar una base, tal que esta sea invertible, para este trabajo y por simplicidad se busca que la matriz base sea una matriz identidad, de tal manera que se evita el trabajo de calcular la inversa de la matriz.

Para que la matriz *B* sea la identidad, se debe seleccionar aquellas variables que aparezcan únicamente en una restricción y estas deben tener un coeficiente igual a uno. Esto se muestra en el ejemplo dado por las expresiones (4.34) y (4.35).

$$\begin{aligned} x_3 + 2x_4 &= 1 \\ 3x_1 + 5x_5 + x_6 &= 2 \\ x_2 - x_4 + 2x_5 + 4x_7 &= 3 \end{aligned} \tag{4.34}$$

Reorganizando las variables y organizando de forma matricial se obtiene:

$$\begin{aligned} x_3 + 2x_4 &= 1 \\ 3x_1 + 5x_5 + x_6 &= 2 \\ x_2 - x_4 + 2x_5 + 4x_7 &= 3 \end{aligned}$$

$$\begin{bmatrix} 0 & 2 & 0 & 0 \\ 3 & 0 & 5 & 0 \\ 0 & -1 & 2 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_4 \\ x_5 \\ x_7 \end{bmatrix} + \begin{bmatrix} 1 & & \\ & 1 & \\ & & 1 \end{bmatrix} \begin{bmatrix} x_3 \\ x_6 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \tag{4.35}$$

$$Nx_N + Bx_B = b$$

De las anteriores expresiones se observa que para obtener la base es necesario acomodar las variables y hacer una búsqueda de aquellas que cumplan con las condiciones antes expuestas. Para este caso se emplea el uso de variables de holgura (para las restricciones de desigualdad) y variables virtuales (en las restricciones de igualdad), como aquellas variables que formarán parte de la base. El uso de variables virtuales tiene como implicación un aumento en las variables del problema, mientras que las variables de holgura son necesarias para convertir las restricciones de desigualdad en restricciones de igualdad, ya que esta es la forma estándar como se plantea el problema (4.1). En la sección posterior se muestra cómo funciona las variables de holgura y las variables virtuales para este trabajo, como se construye la base a partir de estas variables y como se construye el cuadro dual simplex canalizado.

4.7 Variables de holgura y variables virtuales

Las variables de holgura sirven para transformar las restricciones de desigualdad en restricciones de igualdad, la variable de holgura asume el valor que le hace falta a la expresión para llegar a ser igual al límite de la restricción. Esto se ilustra mediante la expresiones (4.36) y la figura (4.37).

$$x_1 + 2x_2 \leq 8 \quad \leftrightarrow \quad x_1 + 2x_2 + x_3 = 8 \tag{4.36}$$

Sí se determina las variables x_1 y x_2:

$$\begin{aligned} x_1 &= 1 \\ x_2 &= 2 \\ 1+4 \leq 8 \quad &\leftrightarrow \quad 1+4+3=8 \end{aligned} \tag{4.37}$$

Se observa que x_3 asume el valor de 3.

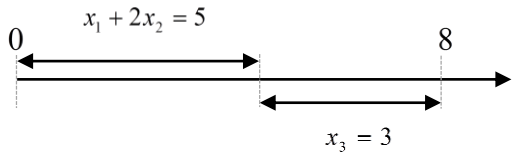


Figura 4.2. Uso de las variables de holgura.

Las variables de holgura deben aparecer sumando en las restricciones del tipo menor igual, mientras que en las restricciones del tipo mayor igual deben aparecer restando, esto con el fin de garantizar que las variables de holgura siempre sean positivas. Un conjunto de restricciones del tipo menor igual se expresa de manera matricial según (4.38).

$$\begin{aligned} Ax &\leq b \\ x &\geq 0 \end{aligned} \quad \leftrightarrow \quad \begin{aligned} Ax + Ix_h &= b \\ x &\geq 0 \\ x_h &\geq 0 \end{aligned} \tag{4.38}$$

Para este trabajo las restricciones del tipo mayor igual serán rescritas según la expresión (4.39).

$$\begin{aligned} Ax &\geq b \\ x &\geq 0 \end{aligned} \quad \leftrightarrow \quad \begin{aligned} Ax - Ix_h &= b \\ x &\geq 0 \\ x_h &\geq 0 \end{aligned} \quad \leftrightarrow \quad \begin{aligned} -Ax + Ix_h &= -b \\ x &\geq 0 \\ x_h &\geq 0 \end{aligned} \tag{4.39}$$

De esta forma se garantiza que el coeficiente de las variables de holgura sea positivo, de tal manera que estas variables forman parte de la base, para así obtener una matriz B identidad.

Para el caso de las restricciones de igualdad se emplea el uso de variables virtuales, esto con el fin de evitar buscar en las restricciones de igualdad aquellas variables que aparecen una sola vez y cuyo coeficiente es la unidad. Esta variable virtual no debe modificar el valor o el efecto de la restricción. La expresión (4.40) muestra el cambio realizado para las restricciones de igualdad.

$$\begin{matrix} Ax = b \\ x \geq 0 \end{matrix} \quad \leftrightarrow \quad \begin{matrix} Ax + Ix_v = b \\ x \geq 0 \\ 0 \geq x_v \geq 0 \end{matrix} \qquad (4.40)$$

Para el método de dual simplex canalizado es de facilidad hacer uso de la expresión (4.40), ya que esta no implica el aumento de las restricciones, sino, un valor en los límites inferior y superior de las variables virtuales; estos límites son cero, de tal manera que la variable virtual siempre obtendrá este valor y se garantiza que la expresión original no es modificada.

4.8 Modelo general de Programación lineal

Teniendo en cuanta lo mencionado en la sección anterior, un modelo cualquiera de programación lineal (4.41) con *n* variables de interés, m_1 restricciones de igualdad, m_2 restricciones del tipo menor o igual y m_3 restricciones del tipo mayor o igual (para un total de *m* restricciones), puede ser rescrito tal como se muestra en la expresión (4.41).

$$\min\{z = cx'\}$$

s.a:

$$\begin{matrix} A_{Ig}x' = b_{Ig} \\ A_{Me}x' \leq b_{Me} \\ A_{Ma}x' \geq b_{Ma} \\ l \leq x' \leq u \end{matrix} \qquad (4.41)$$

Donde:

A_{Ig} : Matriz relacionada con las restricciones de igualdad, de tamaño $m_1 * n$

A_{Me}: Matriz relacionada con las restricciones del tipo menor o igual, de tamaño $m_2 * b$

A_{Ma}: Matriz relacionada con las restricciones del tipo menor o igual, de tamaño $m_3 * n$

$$\min\{z = cx\}$$

s.a:

$$\begin{gathered} A_{Ig}x + I_v x_v = b_{Ig} \\ A_{Me}x + I_{h_1} x_{h_1} = b_{Me} \\ -A_{Ma}x + I_{h_2} x_{h_2} = -b_{Ma} \\ l \le x \le u \\ x_{h_1}, x_{h_2} \ge 0 \\ 0 \le x_v \le 0 \end{gathered} \tag{4.42}$$

Organizando la expresión (4.42) se obtiene:

$$\min\left\{ z = \begin{bmatrix} c & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x' \\ x_v \\ x_{h_1} \\ x_{h_2} \end{bmatrix} \right\}$$

s.a:

$$\overbrace{\begin{bmatrix} A_{Ig} & I_v & & \\ A_{Me} & & I_{h_1} & \\ A_{Ma} & & & I_{h_2} \end{bmatrix}}^{A} \overbrace{\begin{bmatrix} x' \\ x_v \\ x_{h_1} \\ x_{h_2} \end{bmatrix}}^{x} = \overbrace{\begin{bmatrix} b_{Ig} \\ b_{Me} \\ -b_{Ma} \end{bmatrix}}^{b} \tag{4.43}$$

$$\begin{gathered} l \le x' \le u \\ x_{h_1}, x_{h_2} \ge 0 \\ 0 \le x_v \le 0 \end{gathered}$$

Esta expresión es dividida en sus variables básicas y no básicas, de tal manera que las variables básicas están constituidas por las variables virtuales y de holgura, mientras que las variables no

básicas están conformadas por las variables de interés del problema. De tal manera que la base la conforma la matriz identidad que se obtiene debido al uso de las variables de holgura y virtuales.

$$\min\left\{ z = \overset{C_N \; x_N}{[c]} x' + \overset{C_B}{[0]} \overset{x_B}{\begin{bmatrix} x_v \\ x_{h_1} \\ x_{h_2} \end{bmatrix}} \right\}$$

s.a:

(4.44)

$$\overset{N}{\begin{bmatrix} A_{Ig} \\ A_{Ig} \\ A_{Ig} \end{bmatrix}} \overset{x_N}{x'} + \overbrace{\begin{bmatrix} I_v & & \\ & I_{h_1} & \\ & & I_{h_2} \end{bmatrix}}^{B} \overset{x_B}{\begin{bmatrix} x_v \\ x_{h_1} \\ x_{h_2} \end{bmatrix}} = \overbrace{\begin{bmatrix} b_{Ig} \\ b_{Me} \\ -b_{Ma} \end{bmatrix}}^{b}$$

$$l \le x' \le u$$

$$x_{h_1}, x_{h_2} \ge 0$$

$$0 \le x_v \le 0$$

Una vez obtenida la matriz $_B$ y N, es necesario dividir la matriz N en las matrices N_1 y N_2 las cuales están relacionadas con las variables no básicas que se encuentran en su límite inferior y superior respectivamente, la selección de las variables no básicas que deben pertenecer a N_1 o N_2 deben garantizar la condición de cuadro óptimo.

4.9 Construcción del cuadro óptimo

Partiendo del modelo (4.44), se debe generar un cuadro que satisfaga la condición de óptimo, esto es, que la solución pertenezca a un punto extremo del problema, sin embargo dicho punto extremo debe ser infactible, ya que el método dual simplex parte de la infactibilidad y finiquita cuando se encuentra un punto extremo factible, del cual se garantiza que es el óptimo del problema.

Por lo tanto para determinar un punto de arranque se puede construir mediante métodos heurísticos, para cada problema o caso particular, donde se relaje u omitan ciertas restricciones del problema, de tal manera que se obtiene un punto extremo que no pertenece al espacio de soluciones.

Para este caso y por motivos prácticos se omiten todas las restricciones del problema. La metodología es la siguiente:

- Fijar las variables x'_i en su límite superior si $c_i < 0$
- Fijar las variables x'_i en su límite inferior si $c_i \geq 0$
- El resto de variables cuyo coeficiente en la función objetivo son cero, fijarlas en cero

De esta manera se obtiene una solución que es un punto extremo que minimiza la función objetivo y que puede ser infactible ya que no se considera las restricciones del problema, se puede garantizar la condición de opimalidad expuesta en la sección 4.1. Para este caso se tiene que $c_B = 0$, por lo tanto las variables que están en su límite inferior tiene un costo relativo (4.45) menor o igual a cero $(y_{0i} \leq 0)$ y las variables que están en su límite superior presentan un costo relativo (4.46) mayor a cero $(y_{0i} > 0)$.

$$\begin{gathered} c_B B^{-1} N_1 - c_{N_1} = -c_{N_1} \\ y_{0i} = -c_i \\ c_i \geq 0 \end{gathered} \qquad (4.45)$$

$$\begin{gathered} c_B B^{-1} N_2 - c_{N_2} = -c_{N_2} \\ y_{0i} = -c_i \\ c_i < 0 \end{gathered} \qquad (4.46)$$

Adicionalmente en caso de que satisfaga las restricciones del problema, entonces dicha solución es óptima, puesto que se busca en un principio, obtener el valor más bajo de la función objetivo. Como principal desventaja, se tiene la posibilidad de que la solución obtenida este muy lejos del óptimo del problema, por lo tanto el número de iteraciones del método dual simplex canalizado puede aumentar, sin embargo se evita el uso de métodos heurísticos complejos y particulares a cada problema.

4.10 Algoritmo Dual Simplex Canalizado

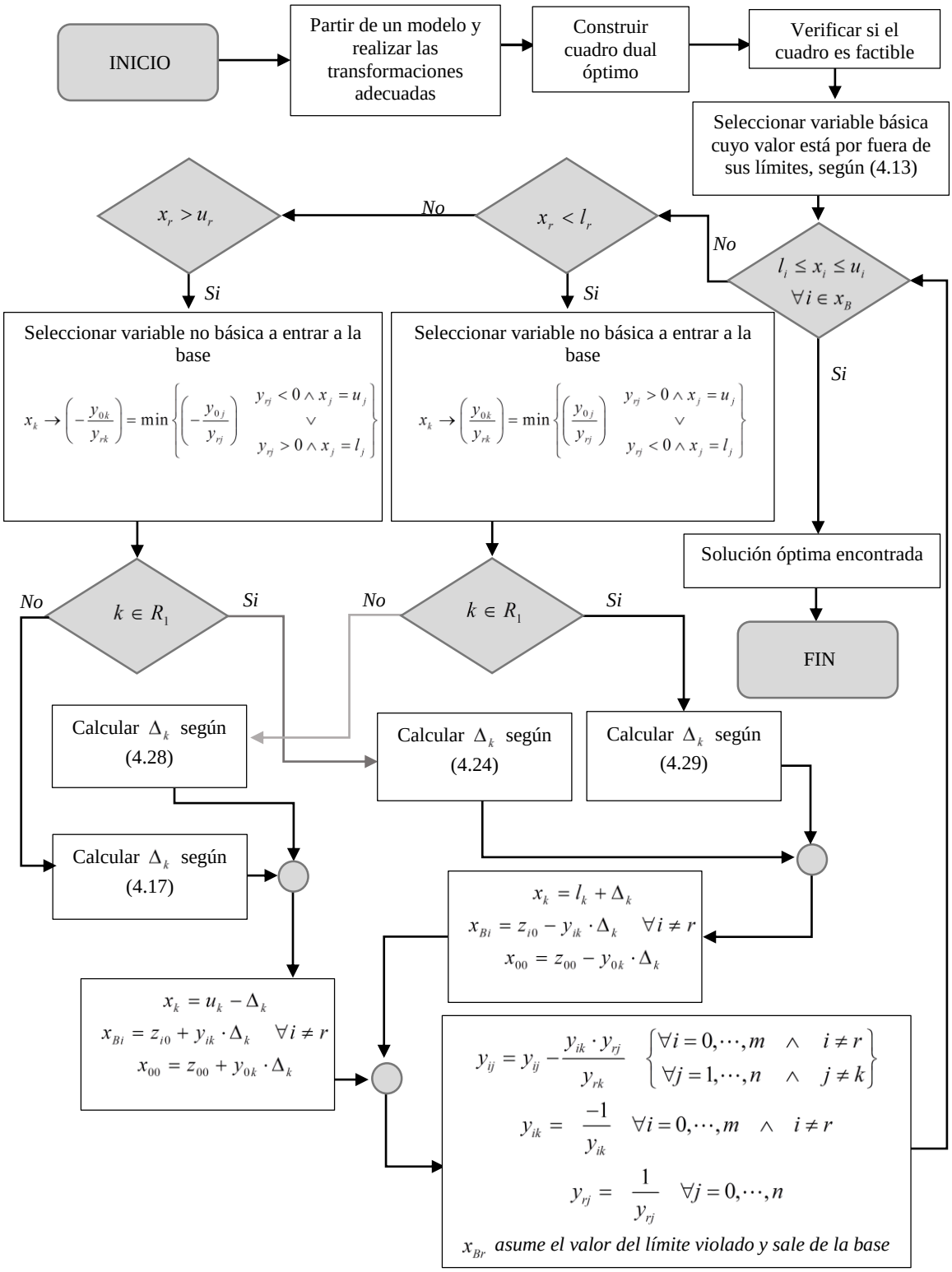


Figura 4.3. Diagrama de flujo Dual Simplex Canalizado.

4.11 Otras consideraciones

En algunas ocasiones se parte de un cuadro simplex inicial el cual es óptimo y factible, sin embargo es posible que se cambien los límites de una o más variables en particular. El interrogante surge en que se debe hacer cuando se cambia los límites de una variable.

Hay 2 opciones que la variable a cuyo límite se modificará pertenece las variables básicas del cuadro actual o la variable pertenece a las variable son básicas del cuadro actual. En el primer de los casos la solución es simple ya que no se modifica el cuadro simplex actual, sólo se debe modificar la cota nueva de la variable y verificar que el valor actual de dicha variable esté dentro de sus nuevos límites, en caso de no estar dentro de dicho límites el cuadro simplex deja de ser factible y por lo tanto se inicia el proceso iterativo del algoritmo dual simplex canalizado.

Para el segundo caso la solución es algo compleja ya que requiere la actualización del cuadro simplex actual, para este caso caben 4 posibilidades:

1. La variable no básica está en su límite superior y se ha modificado su límite inferior
2. La variable no básica está en su límite inferior y se ha modificado su límite superior
3. La variable no básica está en su límite superior, pero este debe ser modificado
4. La variable no básica está en su límite inferior, pero este debe ser modificado

Para los casos 1 y 2 no se hace nada ya que la variable se encuentra en el límite que no ha sido modificado, por lo tanto la solución de este nuevo problema será el mismo.

Mientras que para los casos 3 y 4 se debe actualizar el cuadro simplex de tal manera que la variable asuma el nuevo valor o límite, posteriormente se debe verificar que la solución obtenida al realizar este cambio, siga siendo factible, en caso contrario se debe iterar hasta lograrlo.

Para actualizar el cuadro simplex se parte de una metodología similar a la empleada en la sección 4.3 en la que se modifica el valor de variable no básica en un valor Δ_k. En esta ocasión la variable no básica luego de sufrir el cambio, la función objetivo y las variables básicas sufren un cambio proporcional al valor Δ_k. Haciendo uso de las expresiones (4.22) y (4.23) se obtiene el valor nuevo valor de las variables básicas, la función objetivo se actualiza mediante (4.28), en este caso el valor de Δ_k no es necesariamente positivo ya que depende del cambio de la variable no básica tal como se muestra en la expresión:

$$\Delta_k = L_{Nuevo} - L_{Actual}$$
$$L_{Actual} = x_k \qquad (4.47)$$

Donde:

L_{Nuevo} : Nuevo valor de la variable x_k y del límite inferior l_k sí $k \in R_1$ o u_k sí $k \in R_2$

L_{Actual} : Valor actual de la variable básica x_k , está puede ser u_k sí $k \in R_2$ o l_k sí $k \in R_1$

Adicionalmente se debe actualizar el límite inferior o superior según el caso (3 o 4), tal como se muestra en (4.48).

$$L_{Actual} = L_{Nuevo} \quad \leftrightarrow \quad \left(l_k = L_{Nuevo} \quad \Leftrightarrow \quad k \in R_1\right) \quad \vee \quad \left(u_k = L_{Nuevo} \quad \Leftrightarrow \quad k \in R_2\right) \qquad (4.48)$$

Otra consideración se refiere al caso en que los límites superior o inferior de una variable en particular x_i son iguales, esto es $u_i = l_i$, esto puede suceder para el caso de las variables virtuales o cuando se usan algoritmos como B&B.

Cuando la variable pasa a ser no básica, esta columna puede ser eliminada ya que la variable no entrará de nuevo a la base (Figura 4.4), computacionalmente tiene la ventaja de evitar realizar operaciones sobres la columna asociada a dicha variable, para el caso del método de B&B es posible que el tamaño del cuadro simplex vaya disminuyendo conforme el método avanza.

	RHS	$-x_j$	$-x_{k-1}$	$-x_k$	$-x_{k+1}$	$-x_l$
z	z_{00}	y_{0j}	y_{0k-1}	y_{0k}	y_{0k+1}	y_{0l}
x_1	z_{10}	y_{1j}	y_{1k-1}	y_{1k}	y_{1k+1}	y_{1l}
	⋮	⋮		⋮		⋮
x_i	z_{i0}	y_{ij}	y_{ik-1}	y_{ik}	y_{ik+1}	y_{il}
	⋮	⋮		⋮		⋮
x_m	z_{m0}	y_{mj}	y_{mk-1}	y_{mk}	y_{mk+1}	y_{ml}

→

	RHS	$-x_j$	$-x_{k-1}$	$-x_{k+1}$	$-x_l$
z	z'_{00}	y'_{0j}	y'_{0k-1}	y'_{0k+1}	y'_{0l}
x_1	z'_{10}	y'_{1j}	y'_{1k-1}	y'_{1k+1}	y'_{1l}
	⋮	⋮			⋮
x_k	z'_{k0}	y'_{kj}	y'_{kk-1}	y'_{kk+1}	y'_{kl}
	⋮	⋮			⋮
x_m	z'_{m0}	y'_{mj}	y'_{mk-1}	y'_{mk+1}	y'_{ml}

Figura 4.4. Cambio en el cuadro Dual Simplex Canalizado.

5 Algoritmo de Ramificación y Poda (*Branch and Bound – B&B*)

Es una variante del algoritmo Backtracking, empleado para problemas de programación lineal entera (5.1), el problema consiste en relajar el problema original, permitiendo que las variables enteras asuman valores continuos. Luego se realiza una división del espacio de soluciones (Figura 5.1) generando dos nuevos sub-problemas totalmente disyuntivos y que excluyen la solución relajada del problema antecesor.

$$\min\{z = cx\}$$

s.a:

(5.1)

$$A_{Ig}\, x = b$$
$$l \le x \le u$$
$$x \in \mathbb{Z}$$

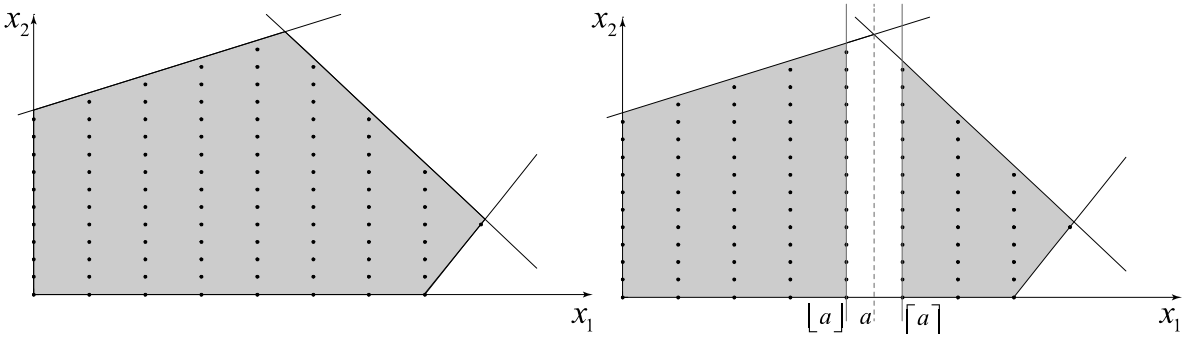


Figura 5.1. División del espacio de soluciones en un problema de 2 variables.

El algoritmo avanza en función de obtener una solución mejor a la conocida hasta el momento. La metodología reduce el espacio de soluciones actualizando la mejor solución conocida hasta el momento (Incumbente). Adicionalmente posee un mecanismo de poda o sondaje que impide que el algoritmo explore por zonas donde la solución obtenida no solución no mejora la incumbente, por lo tanto este método finaliza cuando todos los sub-problemas actuales (o en espera) hayan sido sondados.

El proceso es visto como un grafo cuyo vértice inicial es el problema original, donde la condición de variables enteras ha sido omitida (5.2), los vértices finales de cada rama corresponde a los problemas sondados o podados. En cada problema generado se selecciona una variable x_i , la cual se denomina variable de separación y a través de la cual se hace la división del espacio de soluciones.

$$\min\{z = cx\}$$

s.a: *(5.2)*

$$Ax = b$$
$$l \le x \le u$$

Al obtener la solución del problema inicial, en caso de que esta no sea entera para todas las variables con dicha restricción, se debe generar dos nuevos problemas o sub-problemas (por ser un caso particular del problema original) según la variable x_i seleccionada, (5.3) y (5.4), denominados en este trabajo P^+ y P^- respectivamente.

$$\min\{z = cx\}$$

s.a: *(5.3)*

$$Ax = b$$
$$x_i \ge \lfloor x_i^{k-1} \rfloor + 1$$
$$l \le x \le u$$

$$\min\{z = cx\}$$

s.a: *(5.4)*

$$Ax = b$$
$$x_i \le \lfloor x_i^{k-1} \rfloor$$
$$l \le x \le u$$

A partir de estos 2 problemas a resolver surgen nuevos sub-problemas, esto ocurre hasta que todos los nodos hayan sido sondados. Hasta lo que se ha descrito del Algoritmo B&B surgen varios

interrogantes, ¿cuál es la variable que se debe seleccionar como variable de separación?, ¿cuál de los 2 sub-problemas se debe resolver primero?, ¿a medida que surgen más problemas cual debe ser resuelto?, en las secciones posteriores se trata estos cuestionamientos.

5.1 Ramificación y poda

5.1.1 Ramificación

A medida que se generan 2 sub-probleamas por cada hijo, surge el interrogante en la forma en cómo se debe explorar el árbol generado en el B&B, existe dos metodologías o estrategias populares, para establecer cuales nodos son los próximos a explorar.

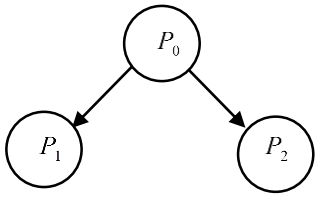
Estrategia FIFO (First In First Out), primero en llegar, primero en salir. Puede verse como una fila de nodos (Lista de nodos en espera) donde el primer nodo de esta fila es atendido (se resuelve el Sub-preblema de dicho nodo). Al ser atendido, dos nuevos nodos se generaran (predecesores del nodo atendido) y llegan a la fila de nodos esperando ser atendidos

Suponer que se parte de un problema inicial (nodo raíz) P_0

Etapa	Lista de nodos en espera
1	P_0

Figura 5.2 Estrategia FIFO etapa 1.

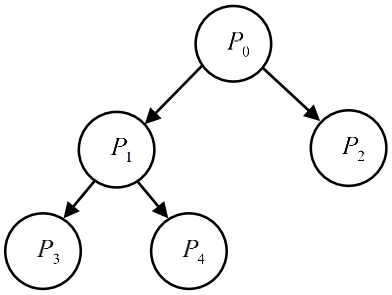
Se define una primera etapa en la cual el único nodo en la fila es P_0 (Figura 5.2), a partir de este se generan dos nuevos problemas de los cuales, el problema P_1 es el primero en llegar en la fila, esto se denomina etapa 2 (Figura 5.3).



Etapa	Lista de nodos en espera	
1	P_0	
2	P_1	P_2

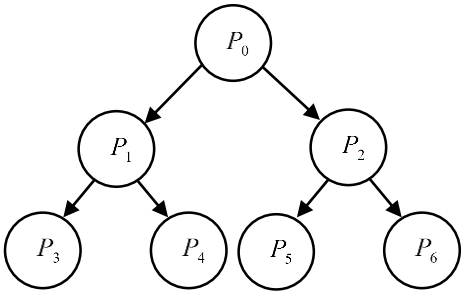
Figura 5.3 Estrategia FIFO etapa 2.

De P_1 se desprende 2 nuevos problemas, P_3 y P_4, los cuales llegan a la fila en el orden mencionado, esto corresponde a la etapa 3 (Figura 5.4). En la etapa siguiente (4) el problema P_2 es atendido y los problemas P_5 y P_6 llegan a la cola de la fila (Figura 5.5). El problema continúa hasta que en la etapa actual todos los nodos hayan sido atendidos, para este ejemplo siempre que un nodo es atendido surgen 2 nuevos problemas, pero es posible que la solución de dicho nodo sea infactible (lo que implica sondaje) y por tanto en dicha etapa no llegarán nuevos nodos en la lista de espera.



Etapa	Lista de nodos en espera		
1	P_0		
2	P_1	P_2	
3	P_2	P_3	P_4

Figura 5.4 Estrategia FIFO etapa3.



Etapa	Lista de nodos en espera			
1	P_0			
2	P_1	P_2		
3	P_2	P_3	P_4	
4	P_3	P_4	P_5	P_6

Figura 5.5 Estrategia FIFO etapa 4.

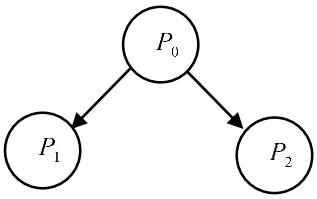
Estrategia LIFO (Last In First Out), último en llegar, primero en salir. Al igual que la estrategia anterior, se puede ver como una fila donde el último nodo de esta fila es atendido. Al ser atendido, dos nuevos nodos se generan (hijos del nodo atendido) y llegan a la fila de nodos en espera, donde el último de estos 2 será atendido.

Al igual que el ejemplo anterior, suponer que se parte de un problema inicial (nodo raíz) P0

Etapa	Lista de nodos en espera
1	P_0

Figura 5.6 Estrategia FIFO etapa 1.

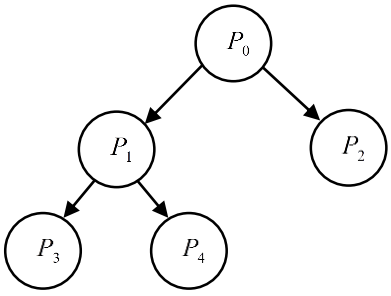
Se define una primera etapa en la cual el único nodo en la fila es P_0 (Figura 5.6), a partir de este se generan dos nuevos problemas de los cuales, el problema P_1 es el primero en llegar en la fila, esto se denomina etapa 2 (Figura 5.7).



Etapa	Lista de nodos en espera	
1	P_0	
2	P_1	P_2

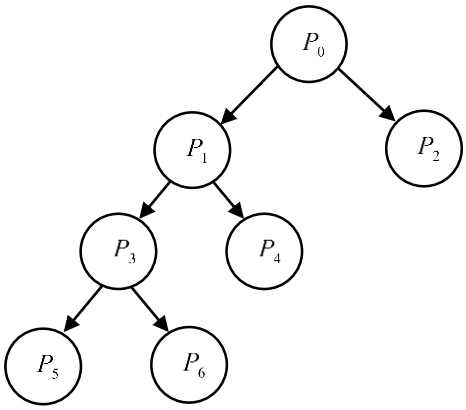
Figura 5.7 Estrategia LIFO etapa 2.

De P_1 se desprende 2 nuevos problemas, P_3 y P_4, los cuales se ubican de primeros en la fila en el orden mencionado, esto corresponde a la etapa 3 (Figura 5.8). En la etapa siguiente (4) el problema P_3 es atendido y los problemas P_5 y P_6 se ubican en la cabeza de la fila (Figura 5.9). El problema continúa hasta que en la etapa actual todos los nodos hayan sido atendidos, para este ejemplo siempre que un nodo es atendido surgen 2 nuevos problemas, pero es posible que la solución de dicho nodo sea infactible (lo que implica sondaje) y por tanto en dicha etapa no llegarán nuevos nodos en la lista de espera o fila.



Etapa	Lista de nodos en espera		
1	P_0		
2	P_1	P_2	
3	P_3	P_4	P_2

Figura 5.8 Estrategia LIFO etapa 3.



Etapa	Lista de nodos en espera			
1	P_0			
2	P_1	P_2		
3	P_3	P_4	P_2	
4	P_5	P_6	P_4	P_2

Figura 5.9 Estrategia LIFO etapa 4.

En este trabajo se implementa la metodología LIFO, por simplicidad en la implementación computacional, al valerse de funciones recursivas que permiten obtener con facilidad el próximo nodo a ser atendido.

A pesar de establecer como se debe generar los problemas, aún siguen los interrogantes. ¿Cuál de los 2 nuevos nodos generados, debe ir primero? y ¿Cuál variable se debe elegir como variable de separación? Como respuesta a estos interrogantes se aclara que no hay una metodología única para establecer la próxima variable y el orden en que van los 2 nuevos sub-problemas generados. Respecto al segundo interrogante se emplea una metodología más compleja basada en la referencia [30], la cual se expone en la sección 5.1.3.

Respecto al primer interrogante, se emplea un indicador de sensibilidad de cada sub-problema, dicho indicador es tomado de la metodología empleada para seleccionar la variable a ramificar (sección 5.1.3). Aquel problema con mejor indicador será resuelto primero.

5.1.2 Criterios de Poda

Hay básicamente 3 criterios de sondaje o poda:

1. Cuando la solución del problema es infactible
2. Cuando la solución del problema es peor que la incumbente
3. Cuando la solución encontrada satisface la condición de las variables enteras

5.1.3 Selección de la Variable de separación

La metodología implementada está basada en las referencias [30] y [31], con algunas variantes en las notaciones con el fin de preservar las empleadas a lo largo de este trabajo y una leve variante en la metodología original.

Seleccionar la variable de separación está relacionado a los indicadores calculados para las soluciones obtenidas en los 2 sub-problemas generados $\left(P_i^+, P_i^-\right)$ luego de ramificar la variable x_i en la metodología de B&B. La variable $Is_{p_i^+}$ corresponde al indicador asociado al sub-problema P_i^+ e $Is_{p_i^-}$ corresponde al indicador asociado al sub-problema P_i^-. El "Score" S_i asociado a dicha variable x_i se cuantifica en función de estos dos sub-problemas o indicadores.

$$S_i\left(Is_{p_i^+}, Is_{p_i^-}\right) = (1-\mu)\min_{IS}\left\{Is_{p_i^+}, Is_{p_i^-}\right\} + \mu\max_{IS}\left\{Is_{p_i^+}, Is_{p_i^-}\right\} \qquad (5.5)$$

Siendo μ un parámetro o factor de peso. Adicionalmente se debe hacer un tratamiento especial cuando alguno de los sub-problemas es infactible, ya que el indicador está asociado a la solución del PL de dicho sub-problema. En este trabajo se opta por el uso de la expresión (5.6), siendo Is el indicador asociado al único sub-problema factible ($Is_{p_i^+}$ o $Is_{p_i^-}$).

$$S_i\left(Is\right) = \mu Is \qquad (5.6)$$

En base a lo anterior, el algoritmo básico para la selección de la variable de separación expuesto en las referencias anteriormente citadas, se muestra a continuación:

1. *Dado* $C = \left\{i \in I \mid x_i \notin \mathbb{R}\right\}$ *como el conjunto de variables candidatas.*
2. *Para todos los candidatos* $i \in C$*, calcule el índice* S_i *(*$S_i \in \mathbb{R}$*).*
3. *Obtenga el índice* k *tal que:* $S_k = \max_{i \in C}\left\{S_i\right\}$.

Diversas metodologías son expuestas en [30] y [31], en la mayor parte de los casos, el factor diferenciador se encuentra en el indicador empleado para calcular el Score. En la Tabla 5.1 se muestra dichas metodología y las referencias donde pueden ser consultadas en mayor detalle.

Método	Referencia
Most infeasible Branching	[32], [33]
Pseudocost Branching	[34]
Strong Branching	[30]
Hybrid strong/pseudocost branching	[30]
Reliability branching	[30]
Best Estimate Methods	[33]
Entropic Branching	[35]
Entropic Lookahead Branching	[35]

Tabla 5.1. Métodos para la selección de la variable de separación.

La metodología implementada en este trabajo está inspirada en "Reability branching", con algunas diferencias en la parte en cómo se actualiza los indicadores. Para este método es necesario mencionar previamente en qué consisten los métodos Strong Branching y Pseudocost Branching.

5.1.3.1 Método de Strong Branching

Esta metodología es implementada por el solver CPLEX [30]. La idea de esta metodología es encontrar la mejor la variable del conjunto de candidatos, respecto a un indicador dado, para ello es necesario conocer la solución de los sub-problemas generados por cada variable candidata.

El mayor inconveniente es el costo computacional, por lo que en algunos casos se emplea un estimado de la función objetivo de cada sub-problema, en vez de obtener la solución exacta de cada problema, adicionalmente también se restringe el número de variables candidatas con el fin de aumentar la velocidad del método. Un estimador de la función objetivo consiste en realizar algunas cuantas iteraciones del método Simplex para el problema obtenido, los autores de [30] afirman que el cambio en la función objetivo en un problema disminuye, conforme el número de la iteración aumenta.

De acuerdo a lo anterior, de dicha metodología surgen 2 parámetros, estos corresponden al número reducido de iteraciones γ del Método Simplex (en este caso Dual Simplex Canalizado) para obtener un estimado de la función objetivo y el número reducido de variables candidatas cuyo conjunto se denomina C' y es un subconjunto de C ($C' \subseteq C$). En [36] se recurre a una estrategia denominada "look ahead" la cual consiste en parar el proceso de evaluación de los sub-problemas (para las variables candidatas C') en caso de no encontrarse un mejor indicador para λ sucesivas variables candidatas.

El número de iteraciones γ sugerido en [30] es 2 veces el promedio de iteraciones empleadas en los PL resueltos hasta el momento, en este trabajo se emplea un número fijo de iteraciones.

5.1.3.2 Pseudo Cost Branching

Este método consiste en determinar cómo varía la función objetivo del problema de acuerdo a la variación de las variables candidatas cuando pasan a ser un valor entero. Algunas variantes del Pseudocosto incluyen el histórico de las veces en que se realiza la ramificación de una variable, la versión de este método es expuesto en [30] y [36].

Partiendo de i como el índice de una variable x_i con parte decimal y relacionada a un PL en particular (sub-problema *P*) con función objetivo z_P, se tiene las expresiones f_i^+ (5.7) (asociado a P^+) y f_i^- (5.8) (asociado a P^-) como las variaciones de dicha variable cuando toman el valor de la parte entera alta $\lceil x_i \rceil$ y baja $\lfloor x_i \rfloor$ respectivamente.

$$f_i^+ = \lceil x_i \rceil - x_i \qquad (5.7)$$

$$f_i^- = x_i - \lfloor x_i \rfloor \qquad (5.8)$$

$z_{P_i^+}$ y $z_{P_i^-}$ representan las funciones objetivos de los sub-problemas P_i^+ y P_i^- respectivamente. La variación de la función objetivo para los dos casos está dado por las expresiones (5.9) y (5.10).

$$\Delta_i^+ = Z_{p_i^+} - Z_p \qquad (5.9)$$

$$\Delta_i^- = Z_{p_i^-} - Z_p \qquad (5.10)$$

La ganancia de la función objetivo o variación de la función objetivo ($\varsigma_i^+, \varsigma_i^-$) respecto a la variación de la variable *i* se muestran en las expresiones (5.11) y (5.12)

$$\varsigma_i^+ = \frac{\Delta_i^+}{f_i^+} \qquad (5.11)$$

$$\varsigma_i^- = \frac{\Delta_i^-}{f_i^-} \qquad (5.12)$$

Algunos autores denominan la anterior expresión como Pseudocosto, hay variantes en esta formulación, algunas de ellas como en este caso incluyen el histórico de los ς_i^+ y ς_i^-. Partiendo de σ_i^+ (σ_i^-) como la suma de todos los ς_i^+ (ς_i^-) asociados a un problema *P* donde se ha seleccionado η_i^+ (η_i^-) veces la variable de separación de índice *i* y adicionalmente el sub-problema P_i^+ (P_i^-) tiene una solución factible del PL. El concepto de Pseudocostos empleado en [30] y [31] consiste en el promedio (ψ_i^+, ψ_i^-) de los términos ς_i^+ y ς_i^- a lo largo de las iteraciones realizadas en el algoritmo B&B (expresiones (5.13) y (5.14)).

$$\psi_i^+ = \frac{\sigma_i^+}{\eta_i^+} \tag{5.13}$$

$$\psi_i^- = \frac{\sigma_i^-}{\eta_i^-} \tag{5.14}$$

Los indicadores son definidos de la siguiente forma:

$$Is_{p_i^+} = \psi_i^+ f_i^+ \tag{5.15}$$

$$Is_{p_i^-} = \psi_i^- f_i^- \tag{5.16}$$

Reemplazando (5.15) y (5.16) en (5.5) se obtiene el "Score" de dicha metodología el cual es denominado Pseudocost Branching y se muestra en la expresión (5.17).

$$S_i\left(\psi_i^+ f_i^+, \psi_i^- f_i^-\right) = (1-\mu)\min\left\{\psi_i^+ f_i^+, \psi_i^- f_i^-\right\} + \mu\max\left\{\psi_i^+ f_i^+, \psi_i^- f_i^-\right\} \tag{5.17}$$

En un principio η_i^+ y η_i^- son iguales a cero, en este caso dichas variables adquieren el valor de 1. Adicionalmente cualquiera de las dos variables puede ser cero si aún no ha ramificado por el sub-problema asociado, en este caso el ψ_i^+ (ψ_i^-) asume el valor promedio de los Pseudocostos de las demás variables y del mismo tipo de sub-problema, esto es $\psi_i^+ = \psi_{promedio}^+$ ($\psi_i^- = \psi_{promedio}^-$). Cuando cualquiera de estas 2 situaciones expuestas ocurre, se dice que la variable *i* es no-inicializada ($i \in I$). Los Pseudocostos de una variable son llamados no-inicializados si están sin inicializar en al menos una dirección [30].

5.1.3.3 Reliability branching

Esta metodología combina el concepto de Strong Branching y Pseudocost Branching. La primera se emplea para obtener un estimado de la función objetivo cuando una variable candidata toma un valor entero, de esta manera se puede calcular un estimado de las expresiones (5.9), (5.10) (Δ_i^+, Δ_i^-), para así inicializar y actualizar los Pseudocostos. Reliability branching no solo usa el concepto de pseudocostos no-inicializados (mencionado en el método de Pseudocost Branching), sino que también emplea el término "poco fiable". Se dice que los Pseudocostos de una variable *i* son poco fiables si se cumple la expresión (5.18), siendo η_{rel} el parámetro de confiabilidad. Para las variables candidatas que cumplen el criterio de no-inicialización o poca fiabilidad, se aplica el método de Strong Branching.

$$\min\left\{\eta_i^+, \eta_i^-\right\} < \eta_{rel} \qquad (5.18)$$

. El algoritmo descrito en [30] se muestra a continuación:

1. *Para todos los candidatos* $i \in C$ *calcule el "score"* $S_i\left(\psi_i^+ f_i^+, \psi_i^- f_i^-\right)$ *expresión (5.17) y ordenarlos en orden decreciente de acuerdo a este valor. Para los candidatos* $i \in C$ *que cumplen con (5.18), aplicar los siguientes pasos:*
2. *Obtenga el número de iteraciones* γ *del dual simplex para cada sub-problema* P_i^+ *y* P_i^-*, para obtener los valores* Δ_i^+ *y* Δ_i^- *(con el uso de las expresiones (5.9) y (5.10)).*
3. *Actualice los Pseudocostos* ψ_i^+ *y* ψ_i^- *con los valores estimados de* Δ_i^+ *y* Δ_i^-.
4. *Calcule el Score (5.5) asociado a las ganancias (*Δ_i^+ *y* Δ_i^-*) de las funciones objetivo (5.19).*

$$S_i = S_i\left(\Delta_i^+, \Delta_i^-\right) \qquad (5.19)$$

5. *Obtenga el máximo* S_j *calculado hasta el momento (5.20).*

$$S_j = \max_{i \in C}\left\{S_i\right\} \qquad (5.20)$$

6. *Si el máximo Score determinado en el paso anterior no cambia para* λ *sucesivas variables candidatas evaluadas (tal como en Strong branching), entonces retorne j como el índice de la variable de selección.*

5.1.3.4 Algoritmo implementado

El algoritmo implementado basado en el anteriormente descrito es el siguiente:

1. *Inicialice los siguientes parámetros para la variable k (todas las variables del problema con restricción entera):*

$$\eta_k^+ = 0 \quad \eta_k^- = 0$$
$$\sigma_k^+ = 0 \quad \sigma_k^- = 0$$
$$\psi_k^+ = 1 \quad \psi_k^- = 1$$
$$S_k = -\infty$$

2. *Partiendo del problema actual aplicar Strong Branching para obtener un estimado de la función objetivo para todas las variables candidatas* $i \in C$ *(aquellas variables de restricción entera cuyo valor actual tiene parte decimal), los PL resueltos tienen* γ *iteraciones fijas.*

7. *Con la estimación de la función objetivo (*$\tilde{z}_{P_i^+}$ *y* $\tilde{z}_{P_i^-}$*) obtener los valores* Δ_i^+ *y* Δ_i^- *(expresiones (5.9) y (5.10)).*

8. *Para todos los candidatos* $i \in C$ *calcule el "score"* $S_i\left(\psi_i^+ f_i^+, \psi_i^- f_i^-\right)$ *expresión (5.17) y ordenarlos en orden decreciente de acuerdo a este valor. Para los candidatos* $i \in C$ *ya ordenados y que cumplen con (5.18), aplicar los siguientes pasos:*

9. *Calcule el Score (*S_i*) (5.5) asociado a las ganancias (*Δ_i^+ *y* Δ_i^-*) de las funciones objetivo (5.19). Tener en cuenta la expresión (5.6), en caso de que uno de los sub-problemas es infactible.*

10. *Obtenga el máximo* S_j *calculado hasta el momento (5.20).*

11. *Si el máximo Score determinado en el paso anterior no cambia para* λ *sucesivas variables candidatas evaluadas (tal como en Strong branching), entonces retorne j como el índice de la variable de selección.*

12. *Con la variable de Selección j genere los 2 nuevos sub-problemas (*P_j^+ *y* P_j^-*) del algoritmo B&B (resolverlos de acuerdo a la regla LIFO), una vez resuelto los problemas obtenga las funciones objetivo de ambos* $z_{P_j^+}$ *y* $z_{P_j^-}$ *.*

13. *Calcular las ganancias reales* Δ_j^+ *y* Δ_j^- *(expresiones (5.9) y (5.10)).*

14. *Actualice los Pseudocostos ψ_j^+ y ψ_j^- con las ganancias reales de Δ_j^+ y Δ_j^-.*
15. *El proceso de selección de la variable es cíclico partiendo de nuevo desde el paso 2. El proceso finaliza con el algoritmo B&B.*

Del anterior algoritmo se resalta que los Pseudocostos no son actualizados con el valor estimado del PL resuelto en Strong Branching, estos son calculados cuando se resuelven los 2 sub-problemas generados en el B&B (pasos 12 a 14), sin embargo al usar la regla LIFO esto implica que uno de los 2 es resuelto mientras que el otro está en la lista de espera, lo que implica que uno de los Pseudocostos es actualizado primero.

Una de las desventajas del algoritmo implementado en este trabajo, es que no se garantiza que el problema llegue a la solución óptima, esto sucede porque las variables de selección están restrictas al parámetro η_{rel}, las experiencias obtenidas en este trabajo revelan que para algunos problemas, todos los nodos del problema fueron sondados sin llegar a una solución entera.

5.2 Metodología

En base a los conceptos mencionados a lo largo de este capítulo, el algoritmo B&B implementado consiste en los siguientes pasos:

1. *Fije la incumbente en el valor inicial más un valor diminuto*

$$Incumbente = Z + \delta$$

2. *Relaje el problema inicial, este será el nodo raíz.*
3. *Resuelva el PL del sub-problema actual P, mediante la metodología del capítulo 4.*
4. *Seleccione la variable de separación j (de acuerdo a la metodología de la sub-sección 5.1.3)*
5. *Para la variable j generar los 2 sub-problemas P_j^+, P_j^- y almacenarlos en la lista de nodos en espera.*
6. *De los 2 sub-problemas generados se selecciona el problema con mayor Pseudocosto, esto es:*

$$\max\left\{\psi_j^+ f_j^+, \psi_j^- f_j^-\right\}$$

   El otro sub-problema generado se agrega a la lista de espera según la regla LIFO). El problema seleccionado será el problema actual P en el próximo ciclo.
7. *Verifique los criterios de sondaje o poda expuestos en 5.1.2.*

8. *Actualizar la incumbente si la solución obtenida es entera para las variables.*
9. *Si todos los nodos de la lista de nodos en espera han sido sondados, entonces, pare y muestre la solución entera obtenida (en caso de tenerla). De lo contrario volver al paso 2.*

6 Metodología Consolidada y Parámetros empleados

Este capítulo tiene como intención mostrar la metodología global, la cual consiste en ejecutar los métodos anteriormente descritos.

El proceso se divide en 2 etapas:

- La etapa 1 consiste en la solución del algoritmo colonia de hormigas, expuesto en la sección 3.8. El cual se vale del espacio de búsqueda establecido en el capítulo 2. El resultado de este es la incumbente inicial de la próxima etapa. Adicionalmente a lo largo de las iteraciones se almacena una lista de rutas, denominado rutas élite. Estas rutas son pre-procesadas, de tal manera que se eliminan rutas iguales, con el fin de evitar restricciones redundantes.
- La etapa 2 consiste en el algoritmo Branch and Bound expuesto en el capítulo 5. Esta etapa toma la lista de rutas élite para encontrar la combinación óptima de dicha lista que satisfaga las condiciones del modelo matemático expuesto en la sub-sección 1.4.3.

Los parámetros empleados se muestran en la Tabla 6.1.

Parámetros empleados	
Metodología de reducción	
Pa	0,1
n_p	30
ACO Modificado	
Número de iteraciones	1000
Número de hormigas	$\sqrt{n}$. siendo *n* el número de clientes
α	2
β	5
γ_0	0,1 para actualizar la feromona y 0,9 Para la parte de evaporación forzada emplear
B&B	
Iteraciones del dual simplex canalizado	$100 * max\{m,n\}$ siendo *m* el número de variables básicas del problema y *n* el número de variables no básicas
γ iteraciones del dual simplex canalizada en Strong Branching	40 en el problema o nodo raíz, 20 para el resto de sub-problemas
λ	4
η_{rel}	8
Incumbente inicial	Incumbente inicial obtenido en ACO Modificado + un valor δ
δ	1
Tolerancia B&B ($T_{B\&B}$)	$1*10^{-10}$
Tolerancia Dual Simplex Canalizado (T_{DSC})	$T_{DSC} = T_{B\&B} * 10^{-2} = 1*10^{-12}$

Tabla 6.1. Tabla de parámetros empleados.

7 Resultados

7.1 Introducción

En este capítulo se exponen las pruebas y resultados obtenidos de la metodología implementada. El algoritmo es probado en un conjunto de instancias de la literatura especializada. En este caso, la información de las instancias y de los mejores resultados obtenidos son consultados de la página web Neo, mostrada en la referencia [37].

7.2 Soluciones Obtenidas

Los resultados de la muestran las soluciones obtenidas en las 2 etapas (*E1*, *E2*). En caso de que el problema en la etapa 2 sea infactible (marcado con * en la tabla) por la cantidad de vehículos (si la solución de la etapa 2 es infactible, es porque el problema es infactible desde la etapa 1), o el algoritmo no converge (marcado con – en la tabla) por la tolerancia del algoritmo, la solución de la etapa 1 es tomada como la *Mejor solución* del método. El *BKS* corresponde a la mejor solución encontrada en la literatura, según la referencia [37]. El error es la diferencia entre el *BKS* y la mejor solución obtenida en el método. El tiempo es medido en cada una de las etapas. Este tiempo no incluye el pre procesamiento de la lista élite donde se eliminan las rutas iguales. Adicionalmente se especifica el número de PL resueltos en el B&B (*NPL*).

Instancia		*E1*			*E2*			*Mejor Solución*	*Error*
i	*Nombre*	*BKS*	*Costo*	*t(s)*	*Costo*	*NPL*	*t(s)*		
1	*A-n32-k5*	784	787,8086949	9,017	787,8086949	1	0	787,8086949	0,485803%
2	*A-n33-k5*	661	668,9187415	8,95	664,8218141	1	0	664,8218141	0,578187%
3	*A-n33-k6*	742	743,0040153	9,987	742,6932624	11	0,172	742,6932624	0,093432%
4	*A-n34-k5*	778	786,4370017	10,267	786,4370017	1	0	786,4370017	1,084448%
5	*A-n36-k5*	799	809,2724518	13,653	809,2724518	3	0,109	809,2724518	1,285664%
6	*A-n37-k5*	669	672,4652202	14,471	672,4652202	1	0	672,4652202	0,517970%
7	*A-n37-k6*	949	960,4750066	13,235	-	21	2,359	960,4750066	1,209168%
8	*A-n38-k5*	730	740,9112921	14,803	740,9112921	1	0	740,9112921	1,494698%
9	*A-n39-k5*	822	868,9275416	14,708	849,4615745	3	0,094	849,4615745	3,340824%
10	*A-n39-k6*	831	847,3327804	15,401	847,3327804	1	0	847,3327804	1,965437%
11	*A-n44-k6*	937	960,4468956	18,464	944,7075916	53	12,563	944,7075916	0,822582%

Instancia		*E1*			*E2*			*Mejor Solución*	*Error*
i	*Nombre*	*BKS*	*Costo*	*t(s)*	*Costo*	*NPL*	*t(s)*		
12	*A-n45-k6*	944	1014,163199	18,606	959,8147168	1	0,015	959,8147168	1,675288%
13	*A-n45-k7*	1146	1194,981158	18,861	1194,981158	5	0,297	1194,981158	4,274098%
14	*A-n46-k7*	914	933,6058182	20,145	918,1274159	1	0	918,1274159	0,451577%
15	*A-n48-k7*	1073	1114,193377	20,895	1087,477774	5	0,563	1087,477774	1,349280%
16	*A-n53-k7*	1010	1056,234746	36,431	-	141	30,297	1056,234746	4,577698%
17	*A-n54-k7*	1167	1190,8481	38,292	1182,848089	1	0,156	1182,848089	1,358020%
18	*A-n55-k9*	1073	1088,427433	34,471	1088,427433	5	0,109	1088,427433	1,437785%
19	*A-n60-k9*	1354	1362,276535	43,373	1358,335332	1	0,109	1358,335332	0,320187%
20	*A-n61-k9*	1034	1067,568605	36,08	*	1	2,641	1067,568605	3,246480%
21	*A-n62-k8*	1288	1349,260911	46,486	1339,057254	35	8,266	1339,057254	3,964073%
22	*A-n63-k9*	1616	1663,743035	45,788	1654,643522	7	2,453	1654,643522	2,391307%
23	*A-n63-k10*	1314	1341,18327	51,088	1337,93265	13	2,594	1337,93265	1,821358%
24	*A-n64-k9*	1401	1466,379681	55,153	1466,203365	15	3,625	1466,203365	4,654059%
25	*A-n65-k9*	1174	1186,661133	45,979	1185,630374	1	0,031	1185,630374	0,990662%
26	*A-n69-k9*	1159	1204,089804	62,267	1201,299974	17	1,719	1201,299974	3,649696%
27	*A-n80-k10*	1763	1837,908595	79,834	1816,119375	5	6,719	1816,119375	3,013010%
28	*E-n22-k4*	375	375,2797871	3,797	375,2797871	1	0,015	375,2797871	0,074610%
29	*E-n23-k3*	569	568,5625011	4,146	568,5625011	1	0,015	568,5625011	-0,076889%
30	*E-n33-k4*	835	842,9898296	8,884	842,9898296	23	0,985	842,9898296	0,956866%
31	*E-n51-k5*	521	541,1870622	28,094	530,4473521	1	0,046	530,4473521	1,813311%
32	*E-n76-k7*	682	711,5564733	68,184	711,5564733	1	0,062	711,5564733	4,333794%
33	*E-n76-k8*	735	782,1158112	63,941	767,709671	7	3,843	767,709671	4,450295%
34	*E-n101-k8*	815	851,6739534	177,904	850,8527553	5	7,406	850,8527553	4,399111%
35	*E-n101-k14*	1071	1150,873156	182,697	1143,345772	1	0,266	1143,345772	6,754974%
36	*B-n50-k8*	1312	1335,002982	40,028	1335,002982	1	0	1335,002982	1,753276%
37	*B-n63-k10*	1496	1570,769988	54,019	1570,769988	1	0,016	1570,769988	4,997994%
38	*B-n78-k10*	1221	1277,238572	91,275	-	225	100,531	1277,238572	4,605944%
39	*F-n45-k4*	724	724,6754586	28,309	724,6754586	7	0,359	724,6754586	0,093295%
40	*F-n72-k4*	237	249,3685452	61,077	246,0239	3	0,516	246,0239	3,807553%
41	*F-n135-k7*	1162	1189,446206	276,733	1182,116078	1	4,484	1182,116078	1,731160%
42	*Kelly01*	5646,46	5632,852531	870,272	5632,852531	1	0,282	5632,852531	-0,240991%

Tabla 7.1. Soluciones obtenidas para las 42 instancias probadas.

Con la instancia A-n61-k9 se obtiene una solución infactible desde la etapa1. Por lo tanto la etapa 2 es infactible. Dicha solución emplea 10 vehículos, cuando el número de rutas a emplear debe ser 9.

La Figura 7.1 ilustra los costos obtenidos para cada instancia *i*. El grafico de los costos obtenidos excluye la instancia *Kelly01* por efectos de la escala del gráfico. La Figura 7.2 ilustra el error obtenido para todas las instancias

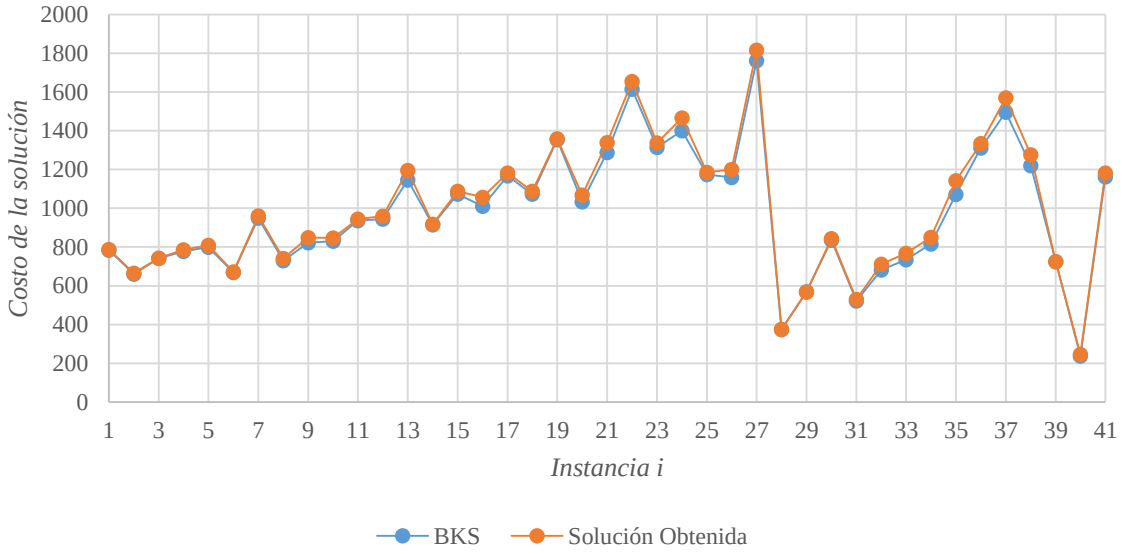


Figura 7.1. Costos obtenidos para las instancias, en comparación con los costos del BKS.

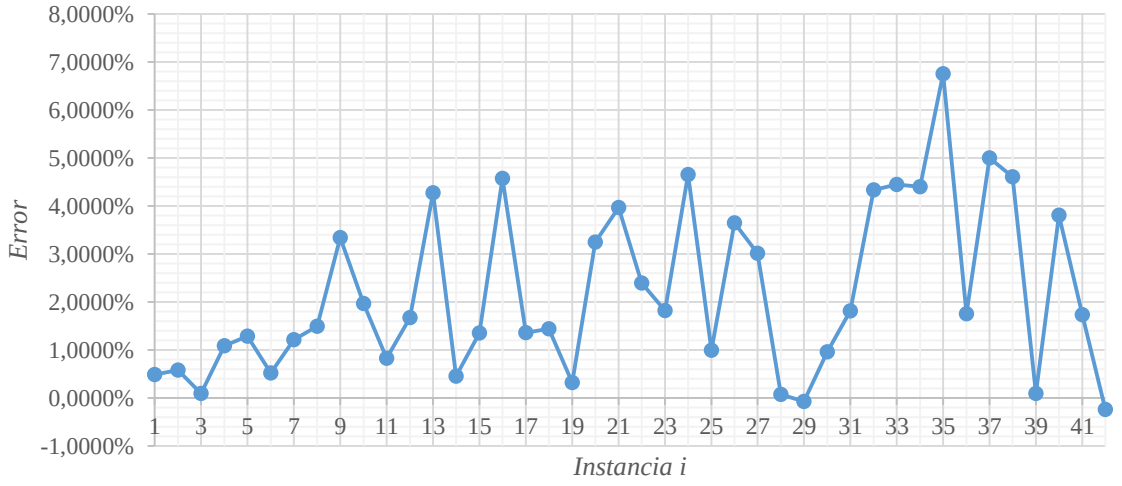


Figura 7.2. Error obtenido para las instancias de prueba.

De los anterior se tiene que el error promedio es de 2,1787%, con un máximo error de 6,7550% y mínimo de -0,2410%.

Algunas soluciones son ilustradas desde la Figura 7.3 hasta la Figura 7.5.

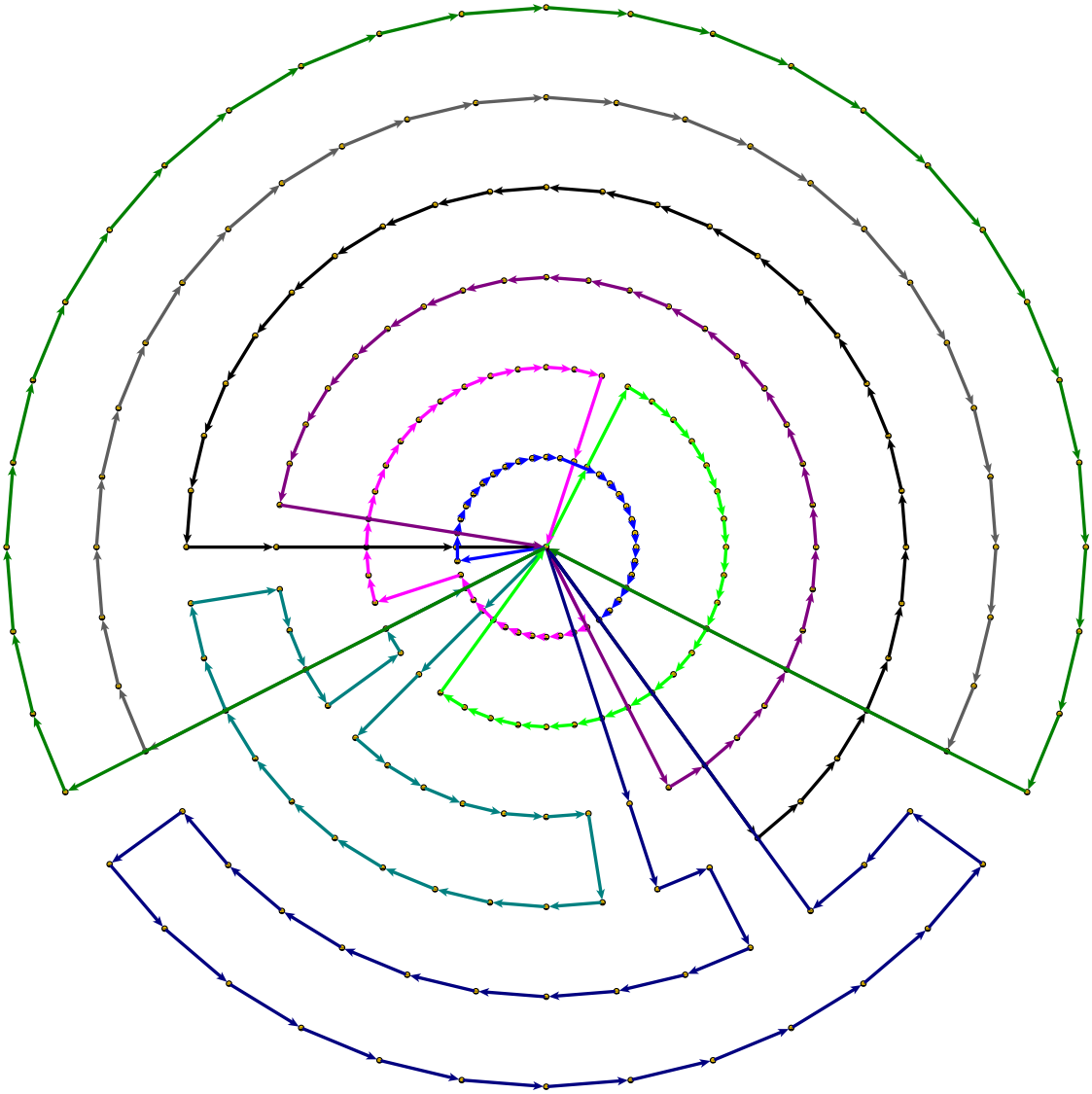
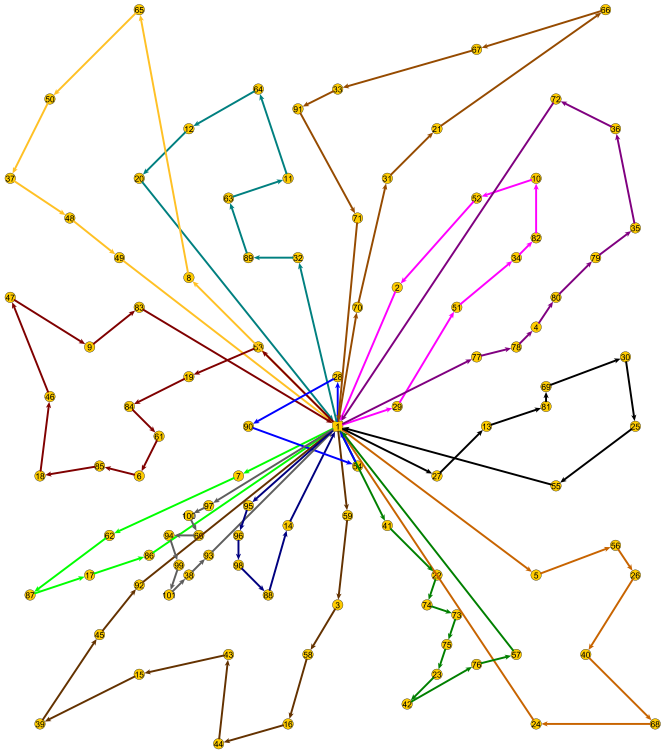
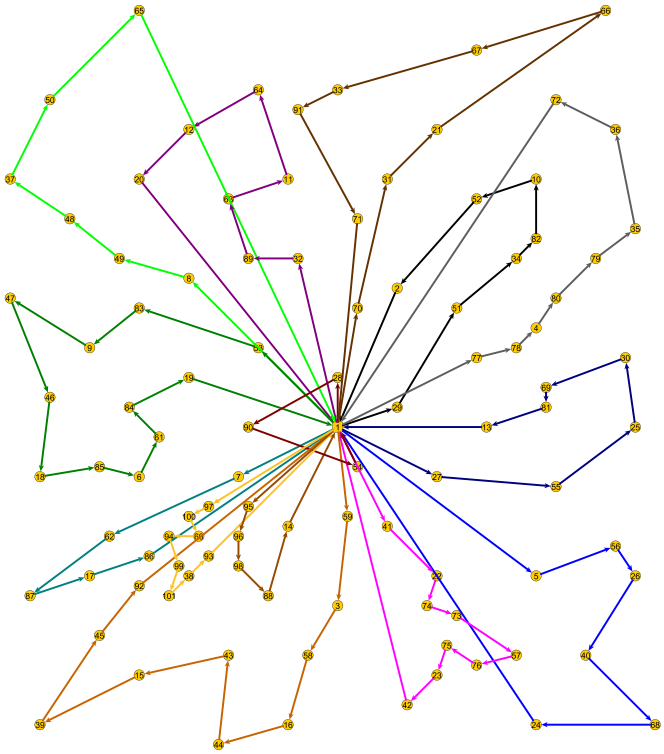


Figura 7.3. Solución obtenida para la instancia Kelly01.

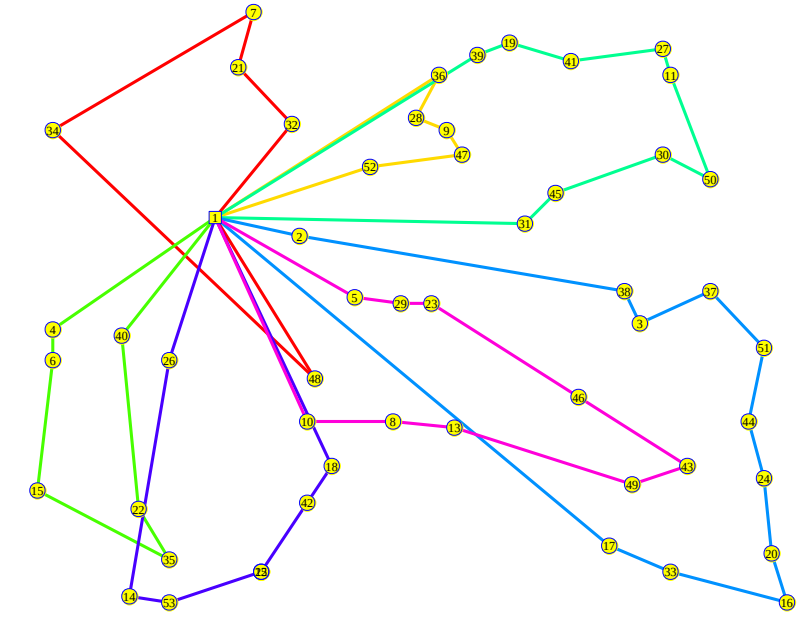


(a)

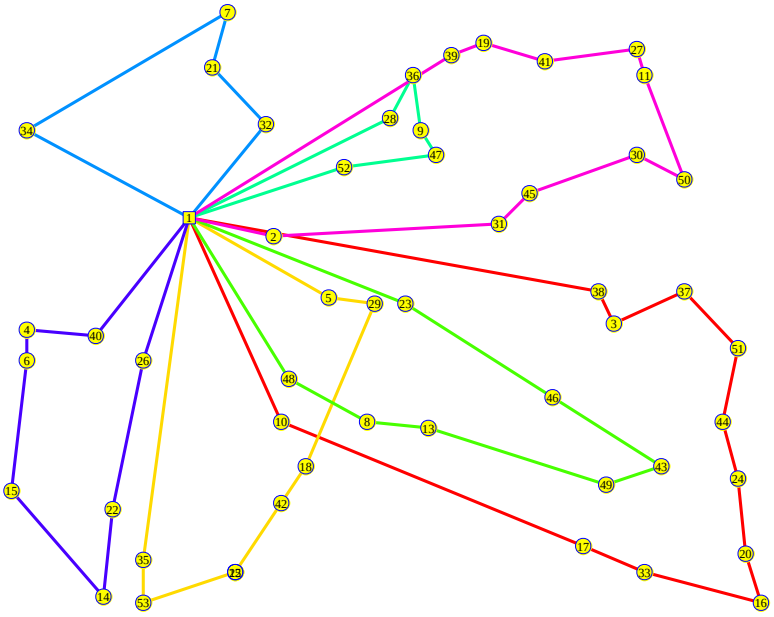


(b)

Figura 7.4. Solución obtenida para la instancia E-n101-k14. (a) Etapa 1. (b) Etapa 2.



(a)



(b)

Figura 7.5. Solución obtenida para la instancia A-n53-k7. (a) Solución obtenida. (b) BKS.

8 Conclusiones, comentarios y trabajos futuros

- Se observa que la diversidad empleada en la lista élite acelera la convergencia del algoritmo B&B
- De igual manera se recomienda eliminar rutas iguales en la lista élite, ya que esto genera restricciones redundantes, que influyen en el método simplex
- Como trabajo futuro, se puede trabajar métodos aproximados desde el método Simplex que generen soluciones factibles de buena calidad
- Se debe tomar un valor de tolerancia en el método Simplex, dicha tolerancia impide que seleccionen elementos cercanos a cero como elementos pivote. El método falla al no trabajar con dicha tolerancia
- La búsqueda local aporta mejoras significativas en la solución obtenida en la exploración realizada por la hormiga
- El método de reducción del espacio de búsqueda posee solo dos parámetros, lo que facilita realizar pruebas variando dichos parámetros, para elegir un par, de acuerdo a un indicador de desempeño
- Se debe tener especial cuidado con las mejoras locales, ya que estas implican reconstrucción de rutas y actualización de los nuevos costos, esto puede generar errores, que repercuten en el desempeño del algoritmo
- La estrategia LIFO adoptada por el algoritmo B&B es una buena alternativa en la solución de problemas de gran complejidad. Esta estrategia permite el uso de funciones recursivas cada vez que se generan los 2 nuevos sub-problemas, almacenando en la pila de punterosl a dirección de memoria del último salto realizado en el algoritmo B&B.
- Respecto al ítem anterior se debe tener especial cuidado en el manejo de memoria, se recomienda liberar las variables dinámicas empleadas siguiendo una estrategia similar a la regla LIFO, donde las últimas variables en ser creadas deben ser las primeras en ser eliminadas a la hora de liberar memoria. Esto es recomendable, ya que las variables dinámicas son almacenadas en la pila de punteros, una eliminación de variables de forma no ordenada, provoca un reajuste de la información en dicha pila, el resultado final es un volcado por apuntar a direcciones de memorias no validas o incorrectas.

- Como trabajo futuro, se puede emplear la matriz *Mb* para la solución directa de los modelos, mediante las técnicas exactas, cuyo espacio de soluciones está restricto a las condiciones puestas por dicha matriz
- Reemplazar el algoritmo de colonia de hormigas por un algoritmo genético de chu beasley, este genera una población diversa la cual puede ser tomada como la lista de rutas élite
- Se puede emplear múltiples colonias de hormigas cuyo τ puede ser diferente para cada colonia. Esto con el fin de generar soluciones diversas, lo cual es beneficioso para la etapa 2, correspondiente al Set Partitioning.

9 Bibliografía

[1] P. Toth y D. Vigo, The vehicle routing problem, Philadelphia: Society for Industrial and Applied Mathematics, 2001.

[2] J. R. G.B. Dantzig, «The Truck Dispatching Problem,» *Management Sci,* vol. 6, nº 1, pp. 80-91, 1959.

[3] B. Eksioglu, A. V. Vural y A. Reisman, «The vehicle routing problem: A taxonomic review,» *Computers & Industrial Engineering,* vol. 57, nº 4, pp. 1472-1483, Noviembre 2009.

[4] G. U. Clarke y J. W. Wright, «Scheduling of Vehicles from a Central Depot to a Number of Delivery Point,» *Operations research,* vol. 12, nº 4, pp. 568-581, 1964.

[5] N. Christofides, A. Mingozzi y P. Toth, «Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations,» *Mathematical programming,* vol. 20, nº 1, pp. 255-282, 1981.

[6] C. Archetti, N. Bianchessi y M. G. Speranza, «A branch-price-and-cut algorithm for the commodity constrained split delivery vehicle routing problem,» *Computers & Operations Research,* vol. 64, pp. 1-10, Diciembre 2015.

[7] R. Fukasawa, Q. He y Y. Song, «A Branch-Cut-and-Price Algorithm for the Energy Minimization Vehicle Routing Problem,» *Transportation Science,* 2015.

[8] M. Pinninghoff, R. Contreras y C. Pantoja, «A comparison of methods for the vehicle routing problem,» *Computing Conference (CLEI), 2014 XL Latin American,* pp. 1, 8, 15-19 , Sept. 2014.

[9] P. Augerat, J. M. Belenguer, E. Benavent, A. Corberán, D. Naddef y G. Rinaldi, «Computational results with a branch and cut code for the capacitated vehicle routing problem,» *Research Report RR949-M, ARTEMIS-IMAG,* 1995.

[10] Y. Marinakis, «Multiple Phase Neighborhood Search-GRASP for the Capacitated Vehicle Routing Problem,» *Expert Systems with Applications,* vol. 39, nº 8, pp. 6807-6815, 15 Junio 2012.

[11] Y. C. M. KAO, «Solving the CVRP Problem Using a Hybrid PSO Approach,» *Computational Intelligence,* pp. 59-67, 2013.

[12] J. De-gang y H. Dong-mei, «Optimization, A research based on K-means clustering and Artificial Fish-Swarm Algorithm for the Vehicle Routing,» *Natural Computation (ICNC), 2012 Eighth International Conference,* pp. 1141,1145, 29-31, May 2012.

[13] Z. Qiu, S. Yu y C. Xiao, «A Ring Construction Algorithm for Solving,» *Mechatronic Science, Electric Engineering and Computer (MEC), 2011 International Conference on ,* pp. 1849,1852, 19-22, Aug. 2011.

[14] R. He, W. Xu, Y. Wang y W. Zhan, «A Route-Nearest Neighbor Algorithm for Large-Scale Vehicle Routing Problem,» *Intelligent Information Technology and Security Informatics (IITSI), 2010 Third International Symposium,* pp. 390,393, 2-4 April 2010.

[15] R. He, W. Xu, J. Sun y B. Zu, «Balanced K-Means Algorithm for Partitioning Areas in Large-Scale Vehicle Routing Problem,» *Intelligent Information Technology Application, 2009. IITA 2009. Third International Symposium,* vol. 3, pp. 87-90, 21-22 Nov. 2009.

[16] M. Dorigo y T. Stützle, Ant Colony Optimization, MIT Press, 2004.

[17] C. Qi, «An Ant Colony Algorithm with Stochastic Local Search for the VRP,» *Innovative Computing Information and Control, 2008. ICICIC '08. 3rd International Conference,* pp. 464-464, 2008.

[18] W. Geng-Sheng y Y. Yun-Xin, «An Improved Ant Colony Algorithm for VRP Problem,» *Intelligent Information Technology and Security Informatics (IITSI), 2010 Third International Symposium,* pp. 129-133, 2-4 April 2010.

[19] M. Wang, «Hybrid Behavior Ant Colony Algorithm for Vehicle Routing Problem,» de *Computational and Information Sciences (ICCIS), 2012 Fourth International Conference*, 2012.

[20] A. Subramanian, E. Uchoa y L. Satoru Ochi, «A hybrid algorithm for a class of vehicle routing problems,» *Computers & Operations Research,* vol. 40, nº 10, pp. 2519-2531, Octubre 2013.

[21] A. Subramanian, «Heuristic, Exact and Hybrid Approaches for Vehicle,» Tesis Doctoral, Niterói, 2012.

[22] P. H. V. Penna, A. Subramanian y L. Satoru Ochi, «An Iterated Local Search heuristic for the Heterogeneous Fleet Vehicle Routing Problem,» *Journal of Heuristics,* vol. 19, pp. 201-232, 2013.

[23] M. L. Balinski y R. E. Quandt, «On an integer program for a delivery problem,» *Operations Research,* vol. 12, nº 2, pp. 300-304, 1964.

[24] R. A. Gallego Rendón, A. H. Escobar Zuluaga y E. M. Toro Ocampo, TÉCNICAS METAHEURÍSTICAS DE OPTIMIZACIÓN, Pereira: Taller de Publiaciones de la Universidad Tecnológica de Pereira, 2008.

[25] H.-T. Kung, F. Luccio y F. P. Preparata, «On finding the maxima of a set of vectors,» *Journal of the ACM (JACM),* vol. 22, nº 4, pp. 469-476, 1975.

[26] M. Granada Echeverry, ALGORITMOS EVOLUTIVOS Y TÉCNICAS BIOINSPIRADAS, Pererira: Universidad Tecnologíca de Pereira, 2009.

[27] J. L. Deneuborg, S. Aron, S. Goss y J. M. Pasteels, «The self-organizing exploratory pattern of the argentine ant,» *Journal of insect behavior,* vol. 3, nº 2, pp. 159-168, 1990.

[28] R. Gallego Rendón, A. Escobar Zuluaga y R. A. Romero Lázaro, Programación Lineal Entera, Pereira: Universidad Tecnológica de Pereira, 2007.

[29] J. B. Vanderlinde, «Planejamento da expansão de sistemas de transmissão usando algoritmos tipo dual simplex especializados em uma estrutura branch and bound,» *Tesis de Maestría,* 2013.

[30] T. Achterberg, T. Koch y A. Martin, «Branching rules revisited,» *Operations Research Letters,* vol. 33, pp. 42-54, 2005.

[31] S. Murkute, «A Computational study of branching rules for multi-commodity fixed-charged network flow problems,» *Tesis Doctoral. Rochester Institute of Technology,* 2013.

[32] A. Atamtürk y M. W. Savelsbergh, «Integer-programming software,» *Annals of Operations Research,* pp. 67-124, 2005.

[33] A. Atamtürk y M. W. Savelsbergh, «Integer-programming software,» *Annals of Operations Research,* vol. 140, nº 1, pp. 67-124, 2005.

[34] M. Benichou, . J. Gauthier, P. Girodet, G. Hentges, G. Ribiere y O. Vincent, «Experiments in mixed-integer linear programming,» *Mathematical Programming,* vol. 1, nº 1, pp. 76-94, 1971.

[35] A. Gilpin y T. Sandholm, «Information-theoretic approaches to branching in search,» *Discrete Optimization,* vol. 8, nº 2, pp. 147-159, 2011.

[36] A. Martin, Integer programs with block structure, Berlin: Habilitations-Schrift, Technische UniversitXat, 1998.

[37] «NEO,» Networking and Emerging Optimization Research Group, [En línea]. Available: http://neo.lcc.uma.es/vrp/. [Último acceso: 18 10 2015].

[38] A. Subramanian, «Heuristic, Exact and Hybrid Approaches for Vehicle,» Niterói, 2012.